A Memetic Algorithm Based PVT Variation-Aware Robust Transistor Sizing Scheme for Power-Delay Optimal Digital Standard Cell Design

by

Zia Abbas, Salman Ahmed

Report No: IIIT/TR/2019/-1

Centre for VLSI and Embeded Systems Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2019

A Memetic Algorithm based PVT Variation-aware Robust Transistor Sizing Scheme for Power-Delay Optimal Digital Standard Cell Design

Mohammed Salman Ahmed, Zia Abbas
Center for VLSI and Embedded Systems Technologies (CVEST)
International Institute of Information Technology (IIIT) Hyderabad
Hyderabad, India
Email: salman.ahmed@research.iiit.ac.in, zia.abbas@iiit.ac.in

***Abstract*— Higher yield, climbing transistor counts and shrinking dimensions of a single Integrated Circuit (IC) have always been the demands in the fabrication market. However, this increased complexity and miniaturization of transistors present a challenge, due to high critical process variations combined with a ubiquitous presence of temperature and supply voltage variations, to achieve the required specification bounds on the desired performance of the circuits. Since optimization has become a very crucial task in IC design, the paper presents an efficient transistor sizing based optimization technique of the CMOS circuits to achieve low power, high performance and high yield design goals.**

The proposed memetic algorithm judiciously utilizes a threshold based local search procedure to improve convergence in its inherent genetic nature. The algorithm optimizes with the effect of temperature $[-55 \text{ to } 125]^{\circ}C$ and supply voltage $\pm 10\%$ variations and in addition a number of statistically sampled sets generated as Gaussian, Latin Hypercube and Correlation Screened schemes of process variations. The proposed technique is applicable to any technology node and has been tested over several standard single-stage and some complex multi-stage digital circuits designed using a Multi-Gate high-K dielectric (MGK) 22nm CMOS model. The reduction in leakage power with propagation delay goes as high as 53% with 9% respectively, as observed across the various digital circuits.

Keywords—Memetic Algorithm, Transistor Sizing, PVT Variations, Leakage Power, Propagation Delay, CMOS VLSI

I. INTRODUCTION

Leakage power has increased substantially with technology scaling and has become a dominant component of power dissipation, limiting the performance of the circuits. Unless measures are taken, the designed circuits would not be able to keep pace with the highly data intensive modern applications where battery life of the devices is paramount.

There are a number of well-known techniques including the use of sleep transistors, variable threshold CMOS, etc. that would minimize standby leakage. However, one of the most efficient ways to combat this problem is to utilize an algorithm that sizes the transistors of the circuit satisfying the different constraints on delay, area or power, especially in the presence of PVT variations. Also, with increasing transistor counts, it becomes essential to make sure that the optimization process yields quality results within reasonable time. The proposed technique is central to a semi-custom design approach based on standard cells that are tailored to the needs of the designer in accordance with the power-delay optimality achieved.

A number of algorithmic techniques have been proposed over the years targeting the same [1-6], including a class of nature-inspired algorithms, i.e., Simulated Annealing (SA), Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO) etc. SA, being a local search technique, suffers from the problem of premature convergence to a local optimum and on the other hand, ABC converges very slowly involving a greater number of simulations. Recent research in optimization has been focused to go beyond these techniques to incorporate a more adaptive behavior towards the same. As such, memetic algorithms have produced successful results in a variety of domains and they perform better than other evolutionary algorithms [7], [8]. Thus, when it comes to transistor sizing, the algorithm presented in this paper also has considerable more flexibility [9] than recently proposed Interior Search Algorithm (ISA) and Gravitational Search Algorithm (GSA) of [2] to produce sufficient reductions in leakage power without increasing propagation delays.

Despite modern developments, the fabrication process is responsible for a number of random variations, due to mask imperfections, improper alignment etc., manifesting themselves in the form of degraded performances and increased power consumptions of the manufactured circuits. Inter-die variations are of primary concern in the case of digital circuits. Moreover, the process corners are not effective nowadays and the design needs to be verified over the entire range of variations [11]. The proposed algorithm exploits the strengths of a statistical approach by intermittently updating its maintained set of worst-case process samples and is more efficient in this sense, as compared to a conventional worst-case methodology [3].

The paper is organized as follows: In Section II, the optimization algorithm is proposed followed by the results of

978-1-5386-6648-7/19/$31.00 ©2019 IEEE
DOI 10.1109/ICCD46524.2019.00060

optimizing a mirror FA cell at nominal conditions in Section III. Section IV and V discuss the effect of environmental and process variations and optimization results in their presence, respectively. In this paper, a detailed analysis is first presented for a full adder cell and it is later extended to the different cells in Section VI. Section VII presents conclusions.

II. Optimization Algorithm

Multi-objective memetic algorithms improve upon the convergence of the conventional population-based genetic algorithms and help to achieve it faster using heuristic local searching [7], [9] as they mimic a cultural evolution approach. The type of local search procedure and its frequency depends on the landscape presented by the problem [8], [12].

A. Conventions

Some of the conventions followed throughout this paper are as follows. Depending on the context, superscripts are used to differentiate sizings or populations respectively and subscripts are used to refer to elements within i.e., *W* or *L* for sizings and sizings for populations. For example, S_t^y refers to a t^{th} element in a y^{th} sizing consisting of a list of channel widths and lengths of the PMOS and NMOS devices of the circuit under test in the order W_p, W_n, L_p, L_n.The two objectives of *leakage* and *delay* are referred by the sizings (e.g., $y\{leakage\}$ denotes leakage power as calculated by equation (1) when the netlist contains y^{th} sizing). The values of the objectives also vary in the context of nominal or process optimization.

$$leakage = \max_{Samples} \left(\sum_{i=1}^{2^{fan-in}} w_i l_i\right) \quad (1)$$

$$delay = \max_{Samples} \left(\sum_{i=1}^{paths} w_i d_i\right) \quad (2)$$

Here in equations (1) and (2), l_i is the leakage power of circuit for i^{th} input combination, d_i is the propagation delay for i^{th} critical *path* and w_i are individual weights such that $\sum_i w_i = 1$. *Samples* represent the worst-case sample set of the process parameters, if applicable, otherwise only the nominal sample.

In the case of nominal optimization, *leakage* represents average leakage power if the weights are chosen to be equal. In case a particular input combination is less desired as it occurs infrequently then to improve runtime of the algorithm, the corresponding weight can be set to zero or vice versa. In case of complex cells with large *fan-in*, a subset of the total input combinations can be taken. In case of process optimization, the worst-case sample set that is maintained, is updated frequently through as the algorithm proceeds. So *leakage* in this context refers to the maximum average leakage power that occurs across all the worst-case process samples.

B. Threshold based local search algorithm

The local search procedure is inspired by the threshold accepting (TA) algorithm [14]. It is similar to the well-known Simulated Annealing (SA) algorithm. As long as the update in the solution is better or not much worse, TA accepts it whereas SA requires probability calculations sometimes causing very worse solutions to be accepted, which is not a desired characteristic in localized search. Thus, the local search adopted in the proposed algorithm is quite simple, yet efficient.

The algorithm searches for neighboring solutions within a $\pm 5\%$ vicinity of channel length and typically $\pm 20\%$ ($R = 40$) for width for transistors selected at random. The degree of variation in width *R* can be tuned to define the scope of the local search. The number of neighboring solutions examined, *localcnt* is typically chosen as 50[13] but can be adjusted depending on *cellsize*, *popsize* and *generations*. If a sizing is found that reduces leakage with delay being at least the same, it is accepted by the algorithm. The default parameters in the algorithm have been determined as suitable to the transistor sizing application by a number of experiments. The *updateNetlist* and *callHSPICE* functions, as the name suggests, are used to update the netlist with the new sizing values and call the HSPICE simulator, respectively. Similarly, *getLeakage* and *getDelay* functions are used to read the leakage and delay values from the measurement output files, respectively.

The parameter bounds are chosen as $W_{min} = L_{min} = 22nm$, $W_{max} = 880\ nm, L_{max} = 30nm$ and the specification bounds (usually customer specific) on leakage (*lb*) and delay (*db*) are chosen as $\max_i(l_i)$ and $\max_i(d_i)$, respectively at the initial sizing of the circuit.

Algorithm: *localSearch*(S^x,$x\{leakage\}$,$x\{delay\}$)

Input: Sizing subjected to local search, leakage and delay at it.
Output: Local search optimized sizing, leakage and delay at it.

```
1:  while i < localcnt
2:    S^y ← S^x
3:    t ← random(|S^x|)
4:    if t ∈ W then
5:      repeat S_t^y ← S_t^x + S_t^x * (R/2 – random(R))%
6:      until S_t^y ∈ [W_min, W_max]
7:    else
8:      repeat S_t^y ← S_t^x + S_t^x * (5 – random(10))%
9:      until S_t^y ∈ [L_min, L_max]
10:   endif
11:   updateNetlist(S^y)
12:   callHspice( )
13:   (y{leakage},y{lb}) ← getLeakage( )
14:   (y{delay},y{db}) ← getDelay( )
15:   if (y{leakage} < x{leakage}) ∧ (y{delay} ≤ x{delay}) then
16:     S^x ← S^y;
17:   else if y{lb} ∧ y{db} then
18:     S^x ← S^y;
19:     cnt ← cnt + 1
20:     if (cnt > threshold) then
21:       cnt ← 0; lb ← lb – lb * (thr)%
22:     endif
23:   else if process then
24:     cnt2 ← cnt2 + 1
25:     if (cnt2 > threshold) then
26:       cnt2 ← 0; S^y ← predict( ); goto 11
27:     endif
28:   endif
29:   i ← i + 1
30: endwhile
31: return (S^x,x{leakage},x{delay})
```

In order to avoid getting stuck, a *threshold* ($\sim localcnt/2$) is maintained that allows different sizing to be accepted [14], even if the leakage or delay values are not better but as long as they satisfy the specification bounds, e.g., as indicated by flags $y\{lb\}$ and $y\{db\}$ for S^y. When the *threshold* crosses, it is reset after some unimproved sizings are accepted by the algorithm and the leakage specification bounds are tightened, i.e., the leakage bound is reduced by a threshold reduction rate *thr* % ($\sim$10%), so that the algorithm doesn't wander randomly in the search space and is constrained more and more to search for better solutions in terms of leakage improvement.

In case of process optimization, the flag *process* is set indicating it is performed after the nominal optimization, the *predict* function generates locally improved sizings that achieved good optimization at the nominal case (obtained at regular checkpoints during the nominal run) that may or may not work well at the worst cases of the process parameters. However, a quick check performed can be used to speed up process optimization [15]. It is executed based on the same *threshold* condition and only when no better sizing can be found conventionally.

C. Memetic algorithm

Usually memetic algorithms start off with a random population. However, an initial sizing that equalizes rise and fall time delays is a better choice as seed to the algorithm than an entirely random sizing [9], [16]. Doing so does not cause premature convergence, the genetic part of the algorithm sees to that. The local search is performed as the initial step *nls* (number of local searches) times to generate sizings as elements of the initial population (*pop*) and random mutations on these fill the remaining *pop*. Its independent searching capability suffices it to be executed only, in case of very large cells to curtail prohibitively excess computation time especially while performing a statistical *process* optimization. On the other hand, the genetic part of the algorithm ensures achievement of a global optimum, provided it is run for a sufficient *generations*. And when the algorithm ends, a power-delay optimal set is extracted from the final population by suppressing its redundancy.

The *fitness* assignment is the inverted normalized domination count [17], the number of times an element of the population (i.e., a sizing) is dominated by others. In other words, an element with best *fitness* 1 will not be dominated by any elements. Domination is shown as a particular sizing being better, either in terms of leakage or delay than the others. Elitism is important to preserve the best in a population. These elements qualify as *elite* if they have *fitness* higher than an elite fitness level, i.e., *ef* ($\sim$0.95) and then they become eligible for local search for further improvements. Selection strategy is tournament based [8] where a number of elements as defined by the tournament size (*ts* = 10) compete each other based on their *fitness* to enter the mating pool of size (*mps* = 0.8 $*$ *popsize*). The *crossover* and *mutate* functions have their usual role as in a genetic algorithm, with *crossover* combining features by performing random multi-point exchange of *W*/*L* amongst the different sizings and *mutate* replacing a *W*/*L* with a new value satisfying the parameter bounds.

The number of *elite* members decreases if *ef* is very strict, in that case to complete the population, some randomly mutated sizings are added, to also preserve the diversity of search. For all the new sizings defined, the simulator is called.

In the case of process optimization, *verifySamples* function is used to check whether leakage and delay corresponding to all the process samples lie below the worst-cases. If not, the particular sample is added to the worst-case sample set *Samples*. This is done at a frequency *v* to curb excess time also ensuring optimization to occur over the entire chosen sample distribution.

Algorithm: *memetic*(S^x,$x\{leakage\}$,$x\{delay\}$)

Input: Initial sizing, leakage and delay at it.

Output: A set of optimal sizing trade-offs

1: **repeat** (*nls*)
2: (S^x,$x\{leakage\}$,$x\{delay\}$) ← *localSearch*(S^x,$x\{leakage\}$,$x\{delay\}$)
3: $pop^i \leftarrow pop^i \cup S^x$
4: **if** *process* ∧ ¬(*nls* **mod** *v*)
5: *verifySamples*( )
6: **endif**
7: **endrepeat**
8: **repeat** (*popsize* − *nls*)
9: $t \leftarrow random(|pop^i|)$
10: $S^y \leftarrow mutate(pop_t^i)$
11: $pop^i \leftarrow pop^i \cup S^y$
12: *updateNetlist*(S^y)
13: *callHspice*( )
14: $y\{leakage\} \leftarrow getLeakage()$
15: $y\{delay\} \leftarrow getDelay()$
16: **endrepeat**
17: **while** $i \leq generations$ **do**
18: $f^i \leftarrow fitness(pop^i)$
19: $elite^i \subset pop^i \mid \exists s: f_s^i > ef \ \forall\, pop_s^i \in elite^i$
20: **for all** $S^x \in elite^i$ **do**
21: (S^x,$x\{leakage\}$,$x\{delay\}$) ← *localSearch*(S^x,$x\{leakage\}$,$x\{delay\}$)
22: **endfor**
23: $pop^j \leftarrow tournamentSelect(pop^i,mps,ts,popsize,fitness)$
24: **repeat**($|pop^j|/2$)
25: $a,b \leftarrow random(|pop^j|)$
26: $(pop_a^j, pop_b^j) \leftarrow crossover(pop_a^j, pop_b^j)$
27: **endrepeat**
28: $pop^i \leftarrow elite^i \cup pop^j$
29: **repeat**($popsize - |elite^i| - |pop^j|$) **steps** 9-11 **endrepeat**
30: **for all** $S^y \in pop^i \setminus elite^i$ **do steps** 12-15 **endfor**
31: $i \leftarrow i + 1$
32: **endwhile**
33: **return** *optimalSet*($pop^{generations}$)

III. Nominal Optimization of a Full Adder Cell

The nominal supply voltage (V_{DD}) is chosen as 1.0V and the temperature as 30℃. The algorithm is developed in Perl with HSPICE simulator in the loop. On a machine with Intel i7-8550U CPU @ 1.8GHz with 7.5 GB memory, a total computation time of 47.18 min was observed for a 28T Full adder.

Fig. 1(a) shows how with the proposed local optimization alone one can achieve some satisfactory reductions. However as expected, the results of memetic search are much better, as shown in Fig. 1(b).

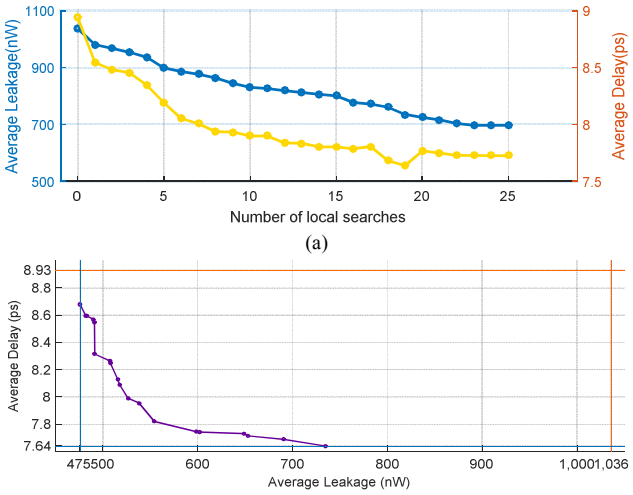
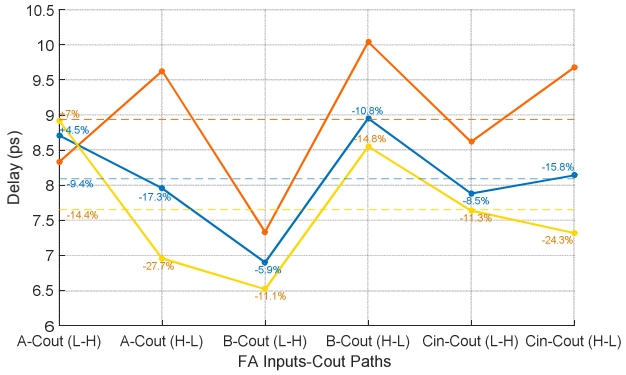


Fig. 1. (a) Initial Local search optimization: Avg. Leakage (nW) -o- and Delay (ps) -o- reduction with different *nls* (b) Memetic search optimization: Pareto optimal leakage-delay curve for a FA cell -o-

Fig. 1(a) also shows how the effects of performing further local search after some runs is a futile effort and thus, *nls* between 15 and 20 is a good choice for the algorithm. Since with each *threshold* crossing, the specifications become stricter, *nls* has to be chosen much smaller, if the reduction in the threshold is much steeper than default *thr* %. Fig. 1(b) reports for a *popsize* = 100 and *generations* = 10, the avg. leakage and delay at different sizings of the optimal set, giving users a lot of options to choose based on their designs.

Fig. 2 shows $Opt.^1$ and $Opt.^2$ are good cases in terms of leakage and delay respectively. With some increase in delay along the path of input A to carry output during a low to high transition, the delays in other paths are fairly optimized. This can be countered by assigning a high weight to the path in the objective function. Leakage power on the other hand, has significantly improved, i.e., 52.6% on average, that too with a simultaneous reduction of 9.4% in delay.

The cut in total power is about 46% from its initial value i.e., 2434.1 nW. The area estimate drawn from the final sizing also features a decline of about 49%. These constraints are optimized without explicitly including them in the optimization process.



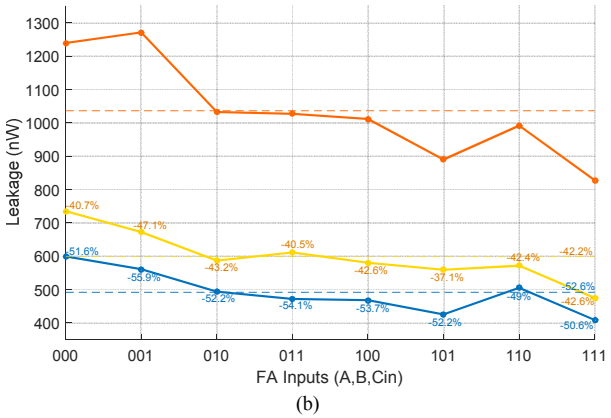


Fig. 2. (a) Propagation Delays (ps) for different paths (b) Leakage (nW) at different input combinations for a FA at Initial Sizing -o-, Opt^1 sizing -o- and Opt^2 sizing -o-

Since the FO4 delay metric is sufficient to track the delay variability well across the PVT corners [18][19], all the simulation results in the paper are obtained at a 4X load equivalent to $C_L = 0.21$fF, where X represents a minimum sized invertor. Fig. 3 demonstrates the effect of varying load for a sizing optimized at 4X works over the shown range. However, at 32X, the propagation delays slightly offset their original values. Therefore, in case of heavy loads, it is desirable to set the specified maximum load before optimization, as the optimized sizings are more or less susceptible at loads above.

IV. Effect of Environmental Varitaions on a FA

Any good sizing methodology should take into consideration the effect of environmental variations [20][21], i.e., supply voltage and temperature effects. The optimization algorithm proposed in the previous section has the capacity to include the dependence of leakage power and delay on these variations to produce a design that works well at a much wider range around the nominal. The supply voltage is varied $\pm 10\%$ around 1.0V in steps of 0.01V and the temperature is varied over the entire standard range as specified for automotive or military applications, i.e., $[-55°C, 125°C]$ in steps of 10°C. Fig. 4 shows this impact on average leakage and delay for the FA cell at initial sizing and after optimization.

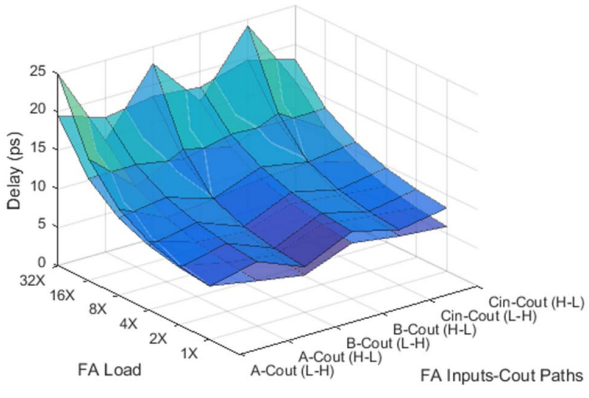
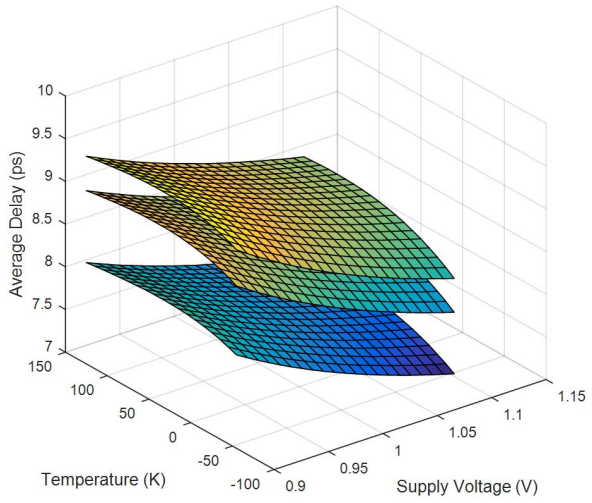
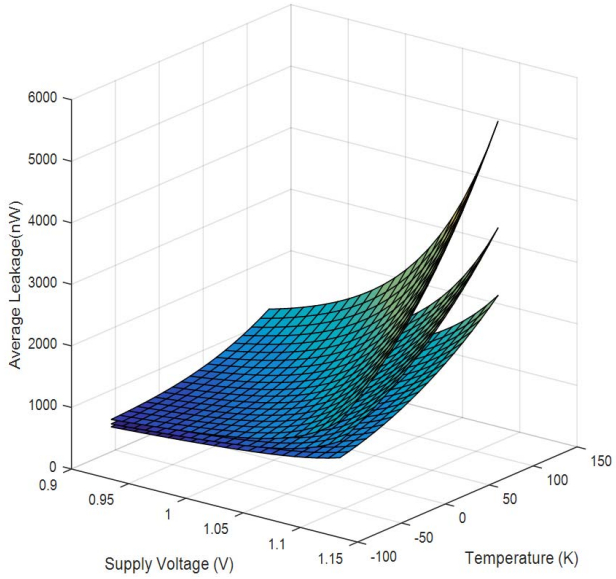


Fig. 3. Effect of load variations on propagation delays (ps) for a FA on Optimized Sizing, Initial Sizing



(a)

Meas. (ps)	μ	σ	σ/μ	WC
$Init.$	8.9152	0.3155	0.0353	9.4322
$Opt.^1$	8.5151	0.3118	0.0366	9.0292
$Opt.^2$	7.7171	0.2829	0.0366	8.1862



(b)

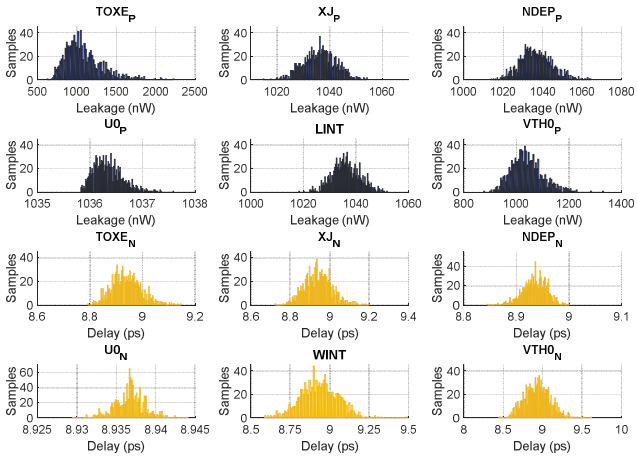
Meas. (nW)	μ	σ	σ/μ	WC
$Init.$	1424.8	869.45	0.6102	5203.2
$Opt.^2$	1038.5	560.79	0.5400	3472.2
$Opt.^1$	822.41	363.30	0.4417	2376.1

Fig. 4. Optimization with the effect of supply voltage and temperature variations on (a) Avg. Delay (b) Avg. Leakage for a FA cell

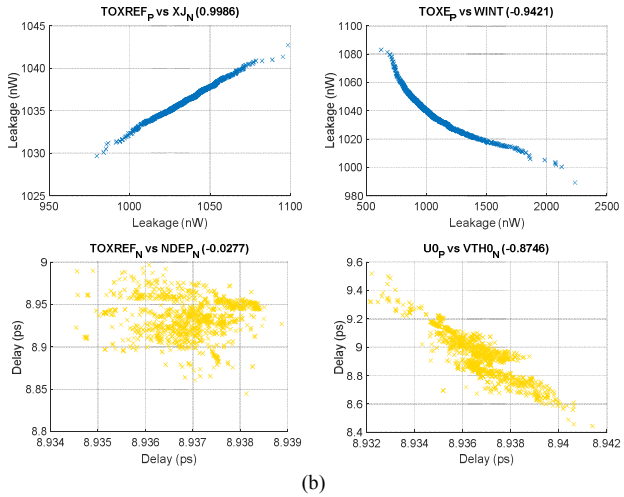
The delay particularly, is insensitive towards changes in temperature and the overall effect on delay is quite less as compared to leakage, which increases by ~5 times. Since the worst-case leakage occurrence is at the highest V_{DD}, $1.1V$ and worst-case delay at 0.9V, the algorithm optimizes by setting these values in the respective netlists to obtain an optimal sizing $Opt.^1$ with leakage reduction of 54.3% and delay reduction of 4.27%. Another trade-off $Opt.^2$ that is favorable in terms of delay, achieves 33.2% and 13.2% reductions in leakage and delay, respectively. Although not observed in delay, there is a significant improvement in terms of variability of leakage as its σ/μ reduces by about 27.6%. However, since the environmental variations occur sporadically during the chip operation, these variations are rarely treated statistically and thus, their worst-case minimization is often sufficient.

V. Process Optimization of a FA

Optimization in the presence of process variations is not a straightforward task, owing to the lack of certainty associated with their occurrence [3], [4], [11], [18]. The proposed optimization algorithm takes into account some of the significant process parameters, i.e., threshold voltage (V_{th}), gate oxide thickness (T_{ox}), junction depth (x_j), channel doping conc. (*ndep*), mobility (u), channel length and offsets (*lint*, *wint*). The individual effects of some of these on leakage and delay are as shown in Fig. 5(a). It is clear now that a combination of all the PVT variations together can cause significant deviations in the specifications.



(a)



(b)

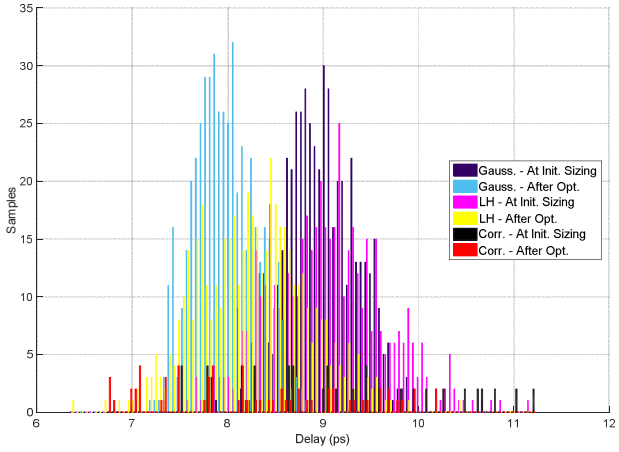
Fig. 5. (a) Distributions of Avg. Leakage and Delay for various process parameters (b) Correlation between some process parameters

The optimization methodology is similar to the nominal case, with certain exceptions. The objective functions are redefined with the effect of process variations as explained in Section II. These variations can be introduced in the simulation process by manipulating the BSIM model parameters in the technology file [10]. *Samples* set is included in the netlist and the cell is simulated at each element of the set with each call.

The variation of the parameters can be modeled with a number of sampling techniques [11] i.e., Gaussian, Latin Hypercube (LH), etc. The maximum variation in each parameter is assumed to be within $\pm 10\%$ of their nominal value and thus, in the case of Gaussian sampling, the 3σ point lies there. The algorithm has the *verifySamples* function to update the *Samples* set, during which the cell is tested for a total of 500 samples based on any of the above standard distributions. Since the computation overhead of each such call is around 3min, the frequency v at which it is executed can be set based on the designer constraints.

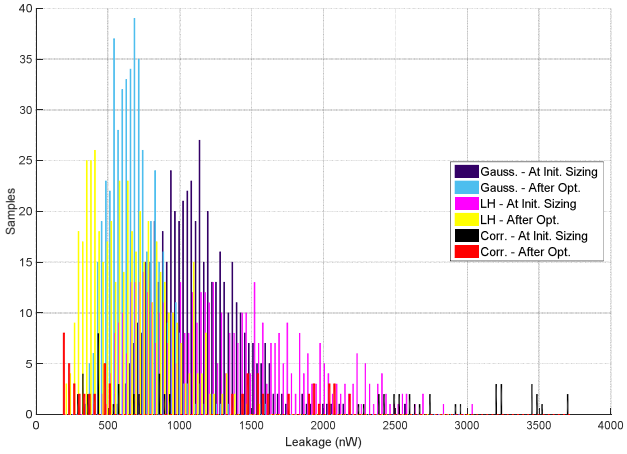
Another simple technique adopted, exploits the correlation between the 17 process parameters considered and the correlation is based on their effect on leakage and delay of the circuit, as shown in Fig. 5(b). The parameters are ordered in terms of the degree of correlation and screened into 6 smaller sets to which a full factorial design is applied to generate 2^6 possible corners of the design. The user can choose between the different techniques with the above one being quicker and Gaussian sampling being one of the relaxed approaches.

Fig. 6 reports the process optimization results for a FA with leakage and delay reductions of 37.4% and 11%, 49.2% and 6.8%, 41.5% and 9.2% at Gaussian, Latin Hypercube and Correlation based sampling, respectively. As observed, there is a significant shift towards left for the samples after optimization (in the graphs of Fig. 6), clearly improving the yield of the circuit.



(a)

Sampling	Meas. (ps)	μ	σ	WC
Gaussian	$Init.$	8.9927	0.3559	9.9014
	$Opt.$	7.9728	0.3236	8.8116
Latin Hypercube	$Init.$	9.0152	0.6130	11.167
	$Opt.$	8.2741	0.5869	10.406
Corr. based full factorial	$Init.$	9.1595	1.0879	11.235
	$Opt.$	8.2728	1.0006	10.195



(b)

Sampling	Meas. (nW)	μ	σ	WC
Gaussian	$Init.$	1123.7	279.65	2561.3
	$Opt.$	685.50	175.86	1601.6
Latin Hypercube	$Init.$	1289.8	538.80	3045.7
	$Opt.$	644.90	267.57	1545.3
Corr. based full factorial	$Init.$	1751.46	1240.5	3720.8
	$Opt.$	1026.8	745.99	2175.5

Fig. 6. Optimization with the effect of process parameter variations with different sampling techniques on (a) Avg. Delay (b) Avg. Leakage for a FA cell

TABLE I. $Init.$ AND FINAL $Opt.$ SIZING OF A FA CELL

DP.	$Init.$ (nm)		$Opt.$ (nm)	
	W_p/L_p	W_n/L_n	W_p/L_p	W_n/L_n
M1	88/22	44/22	73/22	44/24
M2	88/22	44/22	63/22	44/23
M3	88/22	44/22	54/22	35/23
M4	88/22	44/22	88/22	23/23
M5	88/22	44/22	46/22	44/22
M6	88/22	44/22	28/22	22/22
M7	88/22	44/22	23/23	24/22
M8	88/22	44/22	25/22	23/23
M9	88/22	44/22	22/22	22/22
M10	132/22	66/22	47/22	28/22
M11	132/22	66/22	54/24	32/23
M12	132/22	66/22	93/22	48/23
M13	88/22	44/22	43/23	38/22
M14	88/22	44/22	54/22	35/22

Table I shows the initial and optimized sizing of a FA with Latin Hypercube sampling, with design parameters (*DP.*) being labelled as in [3]. A variation of L puts a slight burden on the layout designer. However, to meet rigid performance specification bounds, such relaxation on the manufacturing grid is understood.

VI. OPTIMIZATION ON VARIOUS DIGITAL STANDARD CELLS

The proposed technique is verified with a set of std. cells [22], i.e., NOR2, AND2, NOR3, NAND3, AND3, MUX, FA and MULT. (4x4).

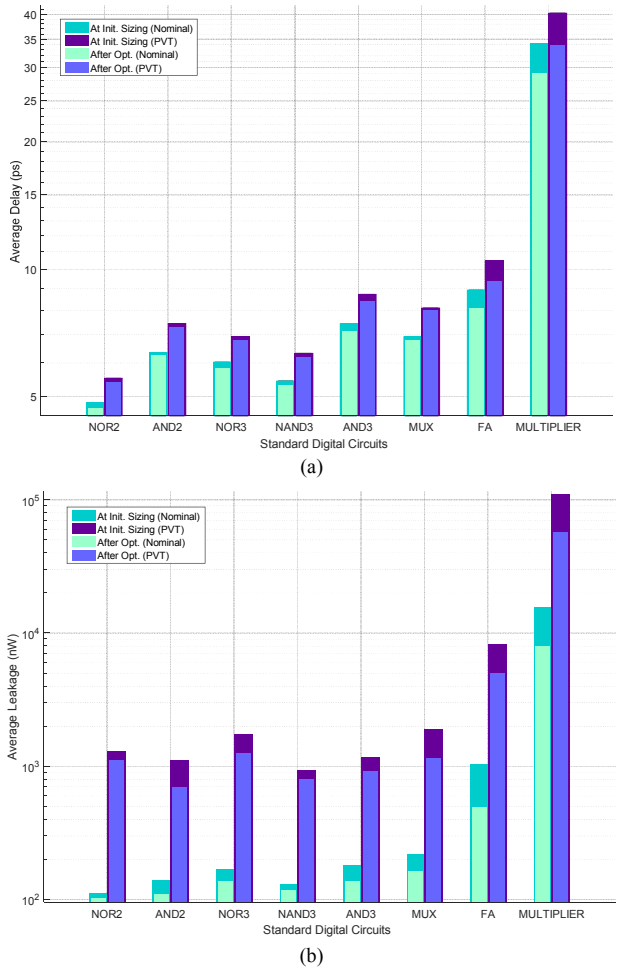


Fig. 7. Optimization at the nominal case and with PVT variations of (a) Avg. Delay (b) Avg. Leakage for a number of standard digital circuits.

Fig. 7 shows the optimization results over this entire set. The results are reported (on a log scale) for best cases of leakage improvements. The multiplier (MULT.), for example, a complex cell, consisting of AND2 and FA cells, shows 48% and 16.3% decrease in leakage and delay, respectively, even with PVT variations.

VII. Conclusions

A brief comparison (shown for a FA cell in Table II) is drawn as to how the proposed technique outperforms other transistor sizing approaches. Although it also performs better than the existing techniques, direct comparisons are not possible in some cases. It is important to note that all the other approaches cannot optimize leakage power without degrading propagation delays of circuit.

Thus, in this paper, an efficient optimization technique has been developed that can help designers to find a robust sizing of transistors exploiting the trade-offs between low power and high performance, to meet specifications even at worst-case operating conditions and as per the yield requirements.

TABLE II. Comparison of Algorithmic Optimization Techniques of Transistor Sizing

Method	SA[5]	ABC[5]	PSO[6]	SQP[3]	Proposed
Avg. Leakage and Delay Improvement	+39.1% −5%	+45% −5.1%	+39.2% -	+58% −5.3%	+52.6% +9.4%
Tech. Node	45nm MGK	45nm MGK	32nm MGK	40nm PDK	22nm MGK
Runtime	~30 min @ 32GB RAM 12cores Xeon	~90 min @ 32GB RAM 12cores Xeon	~60 min @ 32GB RAM 12cores Xeon	-	47.2 min @8GB RAM 4cores i7

References

[1] H. Gupta and B. Ghosh, "Transistor size optimization in digital circuits using ant colony optimization for continuous domain," *International Journal of Circuit Theory and Applications*, vol. 42, no. 6, pp. 642-658, Jun. 2014.

[2] K. Singh, A. Jain, A. Mittal, V. Yadav, A.A. Singh, A. K. Jain, and M. Gupta, "Optimum transistor sizing of CMOS logic circuits using logical effort theory and evolutionary algorithms," *Integration The VLSI Journal*, vol. 60, pp. 25-38, Jan. 2018.

[3] Z. Abbas and M.Olivieri, "Optimal transistor sizing for maximum yield in variation-aware standard cell design," *International Journal of Circuit Theory and Applications*, vol. 44, no. 7, pp. 1400–1424, 2016.

[4] S. Bhardwaj and S. Vrudhula, "Leakage Minimization of Digital Circuits Using Gate Sizing in the Presence of Process Variations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 445-455, Mar. 2008.

[5] P. Gupta, H. Mandadapu, S. Gourishetty and Z. Abbas, "Robust Transistor Sizing for Improved Performances in Digital Circuits using Optimization Algorithms," *20th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2019, pp. 85-91.

[6] P. Gupta, S. Gourishetty, H. Mandadapu and Z. Abbas, "PVT Variations Aware Robust Transistor Sizing for Power-Delay Optimal CMOS Digital Circuit Design," *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, 2019, pp. 1-5.

[7] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 3, pp. 392-403, Aug. 1998.

[8] H. Ishibuchi, T. Yoshida and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204-223, Apr. 2003.

[9] X. Chen, Y. Ong, M. Lim and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591-607, Oct. 2011.

[10] S. K. Saha, "Compact MOSFET Modeling for Process Variability-Aware VLSI Circuit Design," *IEEE Access*, vol. 2, pp. 104-115, 2014.

[11] A. A. Mutlu and M. Rahman, "Statistical methods for the estimation of process variation effects on circuit operation," *IEEE Transactions on Electronics Packaging Manufacturing*, vol. 28, no. 4, pp. 364-375, Oct. 2005.

[12] Yew Soon Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99-110, Apr. 2004.

[13] W. E. Hart, "Adaptive Global Optimization with Local Search," Ph.D. Dissertation, University of California at San Diego, La Jolla, CA, USA.

[14] G. Dueck and T. Scheuer, "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, no. 1, pp. 161-175, Aug. 1990.

[15] G. Yu, W. Dong, Z. Feng and P. Li, "Statistical Static Timing Analysis Considering Process Variation Model Uncertainty," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1880-1890, Oct. 2008.

[16] A. G. Hernandez-Diaz, C. A. Coello Coello, F. Perez, R. Caballero, J. Molina and L. V. Santana-Quintero, "Seeding the initial population of a multi-objective evolutionary algorithm using gradient-based information," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, 2008, pp. 1617-1624.

[17] P. A. N. Bosman and D. Thierens, "Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 259‑289, Nov. 2002.

[18] M. Alioto, E. Consoli and G. Palumbo, "Variations in Nanometer CMOS Flip-Flops: Part I—Impact of Process Variations on Timing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 8, pp. 2035-2043, Aug. 2015.

[19] M. Alioto, G. Scotti and A. Trifiletti, "A Novel Framework to Estimate the Path Delay Variability On the Back of an Envelope via the Fan-Out-of-4 Metric," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 8, pp. 2073-2085, Aug. 2017.

[20] Z. Abbas, M. Olivieri, and A. Ripp, "Yield-driven power-delay-optimal CMOS full-adder design complying with automotive product specifications of PVT variations and NBTI degradations," *Journal of Computational Electronics*, vol. 15, no. 4, pp. 1424–1439, 2016.

[21] Z. Abbas, A. Mastrandrea, and M. Olivieri, "A voltage-based leakage current calculation scheme and its application to nanoscale MOSFET and FinFET standard-cell designs," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2549–2560, 2014

[22] Z. Abbas and M. Olivieri, "Impact of technology scaling on leakage power in nano-scale bulk CMOS digital standard cells," *Microelectronics Journal*, vol. 45, no. 2, pp. 179–195, 2014.