OSFA: A New Paradigm of Gate-Sizing for Power/Performance Optimizations under Multiple Operating Conditions

Subhendu Roy[‡], Derong Liu[‡], Junhyung Um[†], David Z. Pan[‡]
[‡]Department of Electrical and Computer Engineering, University of Texas at Austin, USA
[†]Samsung Semiconductor, Yongin City, Korea (Republic of)
{subhendu,derongliu}@utexas.edu, junhyung.um@samsung.com, dpan@ece.utexas.edu

ABSTRACT

Modern SoCs and microprocessors, e.g., those in smart phones and laptops, typically have multiple operating conditions, such as video streaming, web browsing, standby, and so on. They will have different performance targets and run under different supply voltages. Gate sizing (with threshold voltage assignment) is a fundamental step for power/performance optimization. However, conventional gate sizing algorithms only consider one scenario, e.g., the performance-critical operating condition, which may be over-design for other operating conditions. In this paper, we present a new paradigm of gate sizing, OSFA (One-Size-Fits-All), which performs power/performance optimizations across multiple operating conditions. Based on OSFA, we also adjust the supply voltage targeting overall power optimization. Experimental results on industry-strength benchmarks demonstrate that compared with conventional approach OSFA could provide an average 6.1% reduction in power *without performance loss.*

1. INTRODUCTION

With growing design complexity of System-On-Chip (SoC) and increasing number of cores in microprocessors, same design IP may run under different operating conditions or scenarios [1]. For instance, video streaming and gaming in laptops or smart phones are high-speed applications, whereas the performance requirement for the applications such as web-browsing or text messaging is not stringent. Consequently, supply voltage (V_{dd}) for the performance-relaxed scenarios are typically kept lower to save the dynamic and leakage power.

However, the physical gate sizes of the design need to be *fixed* and discrete across *all* operating conditions. A lot of work have been done in the past on simultaneous gate sizing and threshold voltage (V_{th}) assignment to perform power/performance optimization [2][3][4][5][6][7][8]. But the traditional gate-sizing algorithms consider only one scenario and then designers need to ensure that it meets the timing constraints in all scenarios. This approach has several limitations. Firstly, the timing models in modern cell-libraries are non-linear, and look-up table based [5], and in addition, the supply voltage induced delay scaling in the multi-threshold cell-library depends on V_{th} as well [9]. For instance, the scaled delay at a particular lower V_{dd} would be higher for cells with higher V_{th} than the cells with lower V_{th} as CMOS gate delays depend on $(V_{dd} - V_{th})$. So it may be possible that the gate-sizes suitable for the constrained scenario do not meet the timing constraints for other scenarios under reduced voltage. As a result, designers either need to fix the timing violations incrementally for all scenarios which is tedious or boost up V_{dd} in the scenarios where timing is not met. Secondly, in addition to V_{dd}, the total power of the design depends on (i) the fraction of time spent and (ii) the switching activities of the nets in each scenario. Consequently, consideration of only one scenario during gate-sizing can be sub-optimal in terms of power optimization.

In this paper, we propose a new paradigm of gate-sizing **O**ne-**S**ize-**F**its-**A**ll (OSFA) which selects V_{th} and sizes of the logic gates in the design to optimize power meeting timing constraints across all scenarios. To solve this, we extend the Lagrangian Relaxation based formulation of one scenario to tackle multiple scenarios followed by sensitivity driven power recovery. Multi-threaded implementation is done to cope with the high computational need of our algorithm. A design-space exploration for power vs. V_{dd} is performed to tune V_{dd} in the performance-relaxed scenarios. We also propose a speed-up technique in the design-space exploration for more than two operating conditions. The key contributions of our paper are summarized as follows:

- To the best of our knowledge, this is the first gate-sizing problem formulation considering multiple operating conditions to optimize the total power of any IP design. To tackle multiple operating conditions holistically, scenario aware Lagrangian Relaxation (SALR) problem is formulated for OSFA.
- A cross-layer methodology is developed where system and logic level specifications such as V_{dd}, scenario percentage and the switching activities in different scenarios are considered to select the gate-sizing options which can further guide design-space-exploration by providing feedback to the system level to optimize overall power consumption.
- A speed-up technique is proposed to scale this methodology for higher number of operating conditions, and the degradation in the solution quality for this speed-up is experimentally demonstrated to be small. It is

DAC'15, June 07 - 11, 2015, San Francisco, CA, USA.
Copyright 2015 ACM 978-1-4503-3520-1/15/06 ...$15.00.
http://dx.doi.org/10.1145/2744769.2744885

also observed that the percentage savings in power compared to the conventional methodology increases with the number of operating conditions.

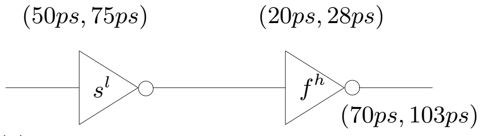
The rest of the paper is organized as follows. Section 2 motivates the problem of gate-sizing under multiple operating conditions. Section 3 presents the problem formulation and OSFA algorithms to solve the scenario aware gate-sizing problem are described in Section 4. Section 5 presents the experimental results for industry-strength large-scale benchmarks with conclusion in Section 6.

2. MOTIVATIONAL EXAMPLES

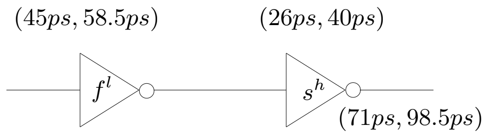
In this section, the problem of scenario aware gate sizing has been motivated by examples from the perspectives of both timing and power. In the example of timing perspective, we have shown that the sizing and threshold voltage assignments which meet the timing in one scenario may not meet the same for the other scenario, and vice versa. On the latter, we have illustrated an example where two sizing schemes may meet timing for both the scenarios, but the scheme considering both scenarios and the scenario percentages (or fraction) results in lesser power than the other scheme which considers only the constrained scenario.

2.1 Timing Perspective

Consider a chain of two inverters with fast and slow scenarios, namely sc_1 and sc_2 respectively with target delays of 70ps and 100ps. Due to lower V_{dd} in sc_2, suppose the delay-scaling factors for sc_2 w.r.t. sc_1 are 1.5 and 1.3 respectively for the slow (high-threshold) and fast (low-threshold) library cells. Fig. 1 illustrates the situation, where the symbols inside the inverters indicate the cell types. For instance, s^l represents a slow and lower size library cell and f^h represents a fast and higher size library cell. The numbers in the bracket indicate the delay of the inverters in sc_1 and sc_2, except for the output it represents the arrival times. It should be noted that the ratio of the delay values in case of second inverter is slightly higher than the scaling factors (1.5 or 1.3) in order to account for the impact of slew degradation at the output of the first inverter.



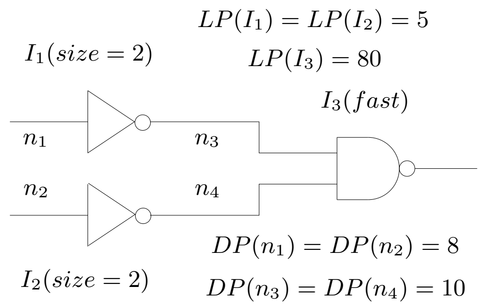
(a) Meeting timing constraints of sc_1, but violating that of sc_2



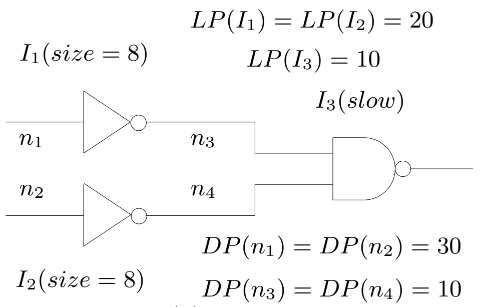
(b) Meeting timing constraints of sc_2, but violating that of sc_1

Figure 1: Motivation for scenario aware gate sizing: timing perspective

We can see that the sizing and V_{th} selection scheme shown in Fig. 1(a) can meet the target delay of sc_1, but violates the delay constraint for sc_2 and vice-versa for the scheme in Fig. 1(b). So by considering only one scenario for gate-sizing, it might not be possible to meet the timing constraints for all the scenarios.



(a) Scheme 1



(b) Scheme 2

Figure 2: Motivation for scenario aware gate sizing: power perspective

2.2 Power Perspective

Consider two scenarios sc_1 and sc_2 and two sizing schemes $Scheme1$ and $Scheme2$ as shown in Fig. 2. Let the supply voltage, usage percentage, and clock frequency for sc_1 are respectively $1V$, 10% and $1GHz$ and those for the other scenario are respectively $0.8V$, 90% and $0.7GHz$. Let sa_{ij} be the switching activity of the net n_i in scenario j and $sa_{11} = sa_{21} = sa_{31} = sa_{41} = 0.5$, $sa_{12} = sa_{22} = sa_{32} = sa_{42} = 0.32$. Suppose both the schemes meet timing in both scenarios. The leakage power (LP) and dynamic power (DP) of the cells and nets, mentioned in Fig. 2(a) and 2(b), are for the scenario sc_1.

If we add up the power numbers in sc_1 the total power in $Scheme1$ and $Scheme2$ are respectively 126 and 130. So $Scheme1$ is the better scheme considering only sc_1. But if we consider both the scenarios, then LP in $Scheme1$ is $0.1\times(5+5+80)+0.9\times(5+5+80)\times(\frac{0.8}{1}) = 73.8$ and DP in $Scheme1$ is $0.1\times(8+8+10+10)+0.9\times((\frac{0.32}{0.5})\times 8+(\frac{0.32}{0.5})\times 8+(\frac{0.32}{0.5})\times 10+(\frac{0.32}{0.5})\times 10)\times(\frac{0.8}{1})^2\times(\frac{0.7}{1}) = 12.9$ totaling 86.7. Note that the scaling factors for V_{dd}, frequency and switching activity of the nets n_i ($\forall i \in [1\ 4]$) in sc_2 w.r.t. sc_1 are respectively 0.8, 0.7 and $\frac{0.32}{0.5}$. Similar calculation on $Scheme2$ gives the total power number as 69.6. So $Scheme2$ is actually the better option when considering both the scenarios.

3. OSFA PROBLEM FORMULATION

Suppose there are n scenarios and each scenario $i \in [1, n]$ is characterized by the voltage level V_{dd}^i, the usage percentage and the switching activities (SA_i) of the nets in the design. The timing targets in each scenario are different and say it is T_i for the i^{th} scenario. In each scenario i, leakage power (LP) depends on V_{dd}^i, and the dynamic power (DP) depends on V_{dd}^i (quadratically) and the switching activities. The formulation for our problem is as follows:

$$\begin{aligned} \text{minimize:} \quad & \sum a_i[LP(V_{dd}^i) + DP(V_{dd}^i, SA_i)] \\ \text{subject to:} \quad & \forall i \in [1, n] \quad T_{delay}(V_{dd}^i) \leq T_i \end{aligned} \tag{1}$$

where, $T_{delay}(V_{dd}^i)$ is the maximum combinational delay between timing start-point to timing end-point in i^{th} scenario and a_i is the fractional percentage for the scenario i so that $\sum a_i = 1$.

4. OSFA ALGORITHMS

We employ a two-step approach to solve the scenario aware gate sizing problem. In the first step, the Lagrangian Relaxation (LR) based formulation in [5] is enhanced to consider more than one scenario. This step gives a solution which meets timing in all scenarios. But since discrete gate sizing problem is NP-hard [10], LR based solution can not be optimal. So a scenario aware sensitivity driven power recovery technique is then applied to further optimize power. Before going into the details of these steps, we first describe the models used for delay and power to consider multiple scenarios.

4.1 Delay and Power Models

Modern industrial cell libraries have look-up table based delay models for various scenarios. In our case, we have taken the industrial benchmarks and cell-library from the recent ISPD'12 contest [11]. However, it contains the delay and leakage power information for single V_{dd}. In Section 5.1, we have described in details how we have generated the scenarios with different voltages. To calculate delay and leakage power across various scenarios, scaling factors have been introduced. Let V_{dd}^{nom} and V_{dd}^i be the supply voltages at the nominal scenario and the i^{th} scenario. Assuming a first-order delay model for CMOS gate delay [9] and velocity saturation constant $\alpha \simeq 1$ the ratio of delay of the i^{th} scenario to that of the nominal scenario is given by:

$$\frac{t_{delay}(V_{dd}^i)}{t_{delay}(V_{dd}^{nom})} = \frac{1 - \frac{V_{th}}{V_{dd}^{nom}}}{1 - \frac{V_{th}}{V_{dd}^i}} \qquad (2)$$

Although V_{dd} has a second-order effect on leakage current [12], we have assumed leakage current to be independent of V_{dd} for the sake of simplicity and thus the corresponding ratio for leakage power is given by:

$$\frac{LP(V_{dd}^i)}{LP(V_{dd}^{nom})} = \frac{V_{dd}^i}{V_{dd}^{nom}} \qquad (3)$$

Dynamic power for a net with switching activity sa is computed as $DP = sa \times f_{clk} \times C_L \times {V_{dd}}^2$, where f_{clk} is the clock frequency and C_L is the total capacitance of the net. Since the internal (short-circuit) power of the library cells are not provided in the library, we have not considered it, but it can be easily added into the power component if available.

4.2 Scenario Aware Lagrangian Relaxation (SALR)

It has been shown that by using Karush-Kuhn-Tucker (KKT) conditions, LR-based formulation for single scenario discrete gate sizing problem can be transformed to the following functional form [5][13]:

$$\alpha \cdot power + \sum_{u \to v} \mu_{u\to v} d_{u \to v} + \sum_{po} \mu_{po}(-r_{po}) + \sum_{pi} \mu_{pi}(a_{pi}) \quad (4)$$

where, $\mu_{u\to v}$, μ_{po} and μ_{pi} are the Lagrange Multipliers (LM) for the timing arc $u \to v$, primary output po and primary input pi respectively, $d_{u\to v}$ is the delay of the arc $u \to v$. r_{po} denotes the required time of arrival at po, a_{pi} denotes the arrival time at pi, and α is the trade-off parameter between power and timing slacks.

To tackle multiple scenarios at a time, we modify the Eqn. (4) as follows:

$$\sum_{i=1}^{n} (\alpha_i a_i [LP(V_{dd}^i) + DP(V_{dd}^i, SA_i)] + \sum_{u \to v} \mu_{u\to v,i} d_{u\to v,i} + \sum_{po} \mu_{po,i}(-r_{po,i}) + \sum_{pi} \mu_{pi,i}(a_{pi,i})) \quad (5)$$

where subscript i has been added in the terms to signify the corresponding terms for i^{th} scenario. It should be stressed that the power components are weighed by the respective a_is, but no such weighing factor is added for the timing terms as the timing needs to be met for all scenarios.

Algorithm 1 SALR Optimization

```
1: Procedure SALROpt(design, library)
2: Initialize Lagrange multipliers for all scenarios;
3: while Leakage power improvement is more than a threshold do
4:     for all $i \in Scenarios$ do
5:         $slackFactor(i) \leftarrow \frac{T_{clk,i}}{T_{clk,i} - worstSlack(i)}$;
6:     end for
7:     $sc \leftarrow$ scenario with the minimum $slackFactor$;
8:     if $worstSlack(sc) > slackThreshold(sc)$ then
9:         $\alpha_{global} \leftarrow \alpha_{global} \times (slackFactor(sc))^2$;
10:    else
11:        $\alpha_{global} \leftarrow \alpha_{global} \times (slackFactor(sc))$;
12:    end if
13:    LRSOpt(design, library, $\alpha_{global}$);
14:    for all $i \in Scenarios$ do
15:        runSTA(design, library, i);
16:        updateLagrangeMultipliers(design, i);
17:    end for
18: end while
19: end Procedure
```

Algorithm 1 presents the key steps of the SALR optimization. At first the maximum load violations are fixed by traversing the cells in reverse topological order and choosing best possible legal cell-types [4]. No max-load violation is introduced throughout the optimization procedure by checking the legality before any cell-type substitution. This is not shown in Algorithm 1. Then the LMs are initialized for all the scenarios (Line 2). This is done by setting the LMs at the timing end-points (primary output/flop input) and then traversing in reverse topological order to assign the multipliers at other pins satisfying the KKT conditions [13]. Then $slackFactor$ for all scenarios are calculated (Line 5) representing the global timing picture of the design across the scenarios. Since lesser the $slackFactor$, more timing constrained the scenario is, the scenario sc with minimum $slackFactor$ is selected to scale α_{global} (Lines 8-12). If the $worstSlack(sc)$ is less than 0, then α_{global} is down-scaled to impose more importance on the timing and vice-versa. If the $worstSlack(sc)$ is greater than $slackThreshold(sc)$, then α_{global} is up-scaled aggressively to impose more weight on leakage power reduction. In our algorithm, $slackThreshold(i)$ is set to be equal to $\frac{T_{clk,i}}{50}$.

At the next step, the Lagrangian sub-problems are solved for individual cells in topological sorted order. For each of the cell, α_i in Eqn. (5) is calculated by scaling α_{global} for individual cell based on the slack of that cell in the i^{th} scenario. The cost for each cell-type (ct) from the cell library is calculated according to Eqn. (6), and ct which minimizes the cost for that cell is selected.

$$cost_c(ct) = \sum_{i=1}^{n}(\alpha_i a_i[LP_{ct}(V_{dd}^i) + \sum_{net \in N}(DP(V_{dd}^i, SA_i))] + \sum_{u \to v} \mu_{u\to v,i}^r d_{u\to v,i}^r + \mu_{u\to v,i}^f d_{u\to v,i}^f) \quad (6)$$

To illustrate this consider the Fig. 3 for cost calculation of the cell c_3. For the timing part of the Eqn. (6), the rising (r) and falling (f) timing arcs ($u \to v$) for the cells, which are immediate fan-ins (c_1, c_2), siblings (c_6, c_7) and fan-outs (c_4, c_5), are taken into account. From the power perspective, leakage power of the cell c_3 with type ct (LP_{ct}) and the dynamic power of the fan-in nets (n_1 and n_2) are considered in the cost computation.

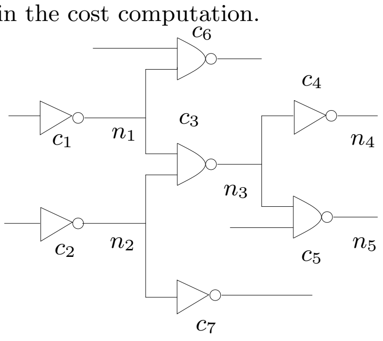


Figure 3: Cost calculation for a cell

Finally, the STA engine is run and Lagrange multipliers are updated at the end of the iteration (Lines 14-17) for all the scenarios. The update is done by first scaling the Lagrange multipliers of individual timing arc/primary output according to the available slack. For instance, the Lagrange multiplier (for rise delay) at primary output (po) is updated as $\mu_{po,i}^r = \mu_{po,i}^r \times \frac{a_{po,i}^r}{T_{clk,i}}$, where $a_{po,i}^r$ represents the rise arrival time at po in the i^{th} scenario. Then the multipliers are updated to match the KKT conditions.

Algorithm 1 is computationally intensive as it needs to run the STA engine and calculate costs across the scenarios. So we have implemented STA (Line 15), update of LMs (Line 16) and cost estimation in 'LRSOpt' using Intel threading building blocks [14]. Typical STA implementation involves the calculation of arrival times (AT) in topological order and required time of arrival (RTA) in reverse topological order. The cells in the design are divided in accordance to the topological levels, and the computation of AT/RTA, LM update and cost estimation in a certain topological level are done in parallel.

4.3 Sensitivity Driven Power Recovery (SDPR)

Since the problem is non-convex, the optimal solution cannot be achieved by only solving the Lagrangian sub-problems. Instead the solution obtained in the first phase is considered as a seed solution on which a sensitivity based power recovery technique is applied to lead towards optimality by recovering more power at the non-critical paths.

In this phase, again another 'while' loop is executed. Algorithm 2 shows the steps of this phase. Like Algorithm 1, the constrained scenario (sc) is determined by choosing the scenario with minimum $slackFactor$. Then the cells are sorted according to its criticality (Line 4), determined by the maximum among the Lagrange multipliers (in sc) of its input pins and then these sorted cells are processed in order, *i.e.*, the cells, which are less timing critical, are processed first (Line 6). For each cell, we calculate a sensitivity factor for each of the available cell-type. The sensitivity factor is the ratio of the power gain to the loss in timing slack by substituting the cell. The cell-type which gives the maximum sensitivity factor is selected.

Algorithm 2 SDPR Optimization

```
1: Procedure SDPROpt($design$, $library$)
2: while Leakage power improvement is more than a threshold do
3:    $sc \leftarrow$ scenario with the minimum $slackFactor$;
4:    Sort cells in accordance to maximum LM in $sc$;
5:    Set all cell status to true;
6:    for all $cell \in sortedCellList$ in increasing order do
7:       if $status(cell) = false$ then
8:          Continue;
9:       end if
10:      Select a celltype maximizing $S_{factor} \leftarrow \frac{\Delta P}{\Delta slack_{loss}}$;
11:      Run BFS in the fan-in/fan-out cone of $cell$;
12:      Set flag to false for all discovered cells;
13:   end for
14:   for all $i \in Scenarios$ do
15:      runSTA($design$, $library$, $i$);
16:      updateLagrangeMultiplier($design$, $i$);
17:   end for
18: end while
19: end Procedure
```

Consider the cell c_3 as shown in Fig. 3. Let its original cell-type be ct_1 and we want to calculate the sensitivity factor for changing its cell-type to ct_2. By changing the cell-type, the input capacitances of c_3 is modified leading to change in input-to-output delays across c_1 and c_2. For each scenario i, the arrival time/slew at n_3 is calculated considering this. Then we calculate the loss in timing slack $\Delta slack_{loss,i}$ as the difference of the updated arrival time and the actual arrival time at n_3. We also consider the impact of change in slew at n_3 by taking the maximum increase in the arrival at the output nets of its fanout cells, *i.e.*, c_4 and c_5, and add that to $\Delta slack_{loss,i}$. If this slack loss is greater than the available slack at n_3 for any scenario, then we skip that cell-type. Otherwise, to consider various scenarios and the rise/fall slack loss, we take the worst case slack loss of the two across all scenarios in sensitivity calculation. Suppose the gain or decrease in power be $\Delta P = \sum_{i=1}^{n} a_i(P_{ct_1,i} - P_{ct_2,i})$ and so we calculate the sensitivity factor ($S_{factor} = \frac{\Delta P}{\Delta slack_{loss}}$) for each of the cell-types available in the library and select the cell-type with maximum S_{factor}.

Once we change the cell-type, we set a flag false corresponding to all the cells which are in fan-in and fan-out cone of c_3 and we do not try to modify the cell types of those cells in that iteration. This process is repeated by going over all cells (note the cells for which the flag becomes false are not processed in that iteration). It might be possible that the timing slack becomes negative for the design in one scenario. This is possible because when we change the cell type we do not propagate the slew impact throughout the design. In such case, we swap the cells, where we find negative timing slack, back to its earlier cell-type (not shown in Algorithm 2). The iterations are continued until we do not get any improvement in leakage power.

5. EXPERIMENTAL RESULTS

We have implemented the algorithms presented in this work in C++ and run it on a Linux machine with 8-Core

2.90GHz CPU and 72GB RAM. In this section, first the test case generation method is described. Then we will experimentally validate that gate sizing considering one scenario may not meet the timing constraints in another scenario and vice versa. Finally, power savings in our algorithm are demonstrated by performing design-space exploration in the gate-sizing step by tuning the system level parameter V_{dd}.

5.1 Test Case Generation

The designs and cell-library for the experimental demonstration are taken from the recent ISPD'12 benchmark suite [11] (fast version). However, the cell-library contains the delay values under one operating condition (one V_{dd}). We have considered V_{dd} for this scenario to be $1.2V$ and created another slow scenario with the timing target equal to 1.5 times that of the nominal scenario. For instance, the timing target for the benchmark 'pci_bridge32_fast' is 660ps, and so the target delay for the slow scenario is 990ps. Eqn.(2) is used to compute the delay values in the slow scenario, and we need the V_{th} values of the cells for this. ISPD'12 cell-library consists of cells with three V_{th} and considering the nominal V_{th} to be around $0.46V$ in [15], we assume those to be $0.4V$, $0.45V$ and $0.5V$. The supply voltage of the slow scenario is varied to search for power optimal solutions across the scenarios. Since the design IPs in laptops or smart phones typically run most of the times under slow operating conditions, we choose the scenario percentage for the fast and slow scenario to be 0.2 and 0.8 respectively.

For industrial designs, the switching activities are captured by VCD/SAIF files which we do not have. So we assume the signal probabilities $SP = 0.5$ for the fast scenario, and random SP for the slow scenario. Then for each case, we generate 500 input vectors maintaining their respective SPs at the primary inputs. Next, we perform 500 Modelsim simulations [16] to compute the signal values at the internal nets and take the average over 500 simulations to obtain the signal probabilities. Finally, switching activity (SA) is computed as $SA = 2 \times SP \times (1 - SP)$ [17]. If switching activities are given instead of assuming randomly, we believe that our algorithm will work with same efficiency, and if not good particularly for dissimilar switching activities across different operating conditions due to the switching activity driven objective function.

5.2 Results

The contest held by ISPD'12 [11] has focused only on leakage power optimization instead of considering both leakage and dynamic power. So it may not be fair to directly compare the power numbers in our approach to the contest winners or other published works based on this. Nevertheless, if we compare the leakage power component of the solutions achieved in our algorithm (in case of single operating condition like the contest), we get 20% (average) lesser leakage power than the contest winner and competitive solution in comparison to the other published works [4][7]. In terms of run-time, [7] is the fastest among all published works and gives solutions for all 14 benchmarks (7 designs with fast and slow version) in 4.9 hours with 2.67GHz CPU, whereas our algorithm can give the same (optimizing the total power) in 3.8 hours and follow an almost linear relationship with the size of the circuits. Since the objective function of our formulation is not the same as that of the ISPD'12 contest, we omit the detailed design-wise comparisons due to space constraints.

To demonstrate the effectiveness of OSFA, we take the benchmark 'pci_bridge32' and run our sizer considering only the fast scenario. Then STA is run for the slow scenario with $V_{dd} = 0.85V$ and we get 1818 timing violations (at the timing-end points, such as primary output or 'D' pin of flip-flops). Then we do the opposite, *i.e.*, size the gates considering the slow scenario and STA is run for the fast scenario. In this case, we get 11 timing violations. So if we just consider single scenario for gate-sizing like the conventional approach, it might not be possible to meet the timing constraints across all the scenarios. This is due to the non-linearity in delay model and non-uniform delay scaling across different V_{th} for a particular V_{dd}, as explained in Section 1. The violations can be fixed by increasing V_{dd}. For instance, if we size considering only fast scenario and then raise V_{dd} of slow scenario to $0.90V$, then it meets the target delays in both scenarios. But this increases the power consumption of the design.

However, by running the OSFA considering both fast and slow scenarios, we can meet the timing constraints in both the scenarios as OSFA has the intelligence to identify which cells are critical in all scenarios and assigns sizes accordingly. This gives us the flexibility in performing design-space exploration. For instance, we set $V_{dd} = 1.2V$ for the fast scenario and then vary V_{dd} of the slow scenario from $0.8V$ to $1.0V$, and run OSFA. Fig. 4 shows the curve for total power of the design vs. V_{dd} of the slow scenario. We can see as V_{dd} increases from $0.8V$, the power consumption initially decreases till $V_{dd} = 0.87V$ and then increases with V_{dd}.

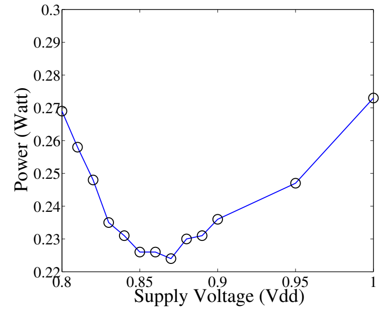


Figure 4: OSFA Design-space exploration by tuning V_{dd} of 2^{nd} scenario for pci_bridge32

The explanation for this behavior is as follows. When V_{dd} is increased, it has two conflicting effects, (i) increase in LP/DP due to its direct V_{dd} dependence, and (ii) decrease in delay of the logic gates facilitating down-sizing or high V_{th} selection resulting in lower LP/DP. If V_{dd} for the slow scenario is too low (such as $0.8V$), slow scenario becomes the constrained scenario and logic gates require up-sizing or low V_{th} selection, and power consumption is high. When it initially increases beyond $0.8V$, slow scenario starts to be less timing-constrained making second effect the prominent one, and decreasing power consumption. But after certain point (here $V_{dd} = 0.87V$), the fast scenario starts to become the constrained scenario. Consequently, second effect becomes submissive because we can not down-size the gates further or select higher V_{th} as the timing constraint of the fast scenario needs to be still met. So beyond this point, first effect plays a dominant role in increasing the power consumption.

Next, we repeat this experiment for all the benchmarks with fixed $V_{dd} = 1.2V$ for the fast scenario and varying V_{dd} of the slow scenario from $0.8V$ to $1.0V$ in steps of $0.05V$ and select the best among all solutions. To compare with the conventional methodology, the sizer is run by considering only fast scenario, and V_{dd} of the slow scenario is bumped up also in steps of $0.05V$ until the timing constraint for the slow scenario is met. Then we compare the obtained power numbers with that achieved by the design-space exploration.

Table 1 presents the comparison for all the benchmarks in terms of power. Column 2 shows the number of cells in the design. Columns 3 and 4 present the total power in the conventional and OSFA methodology respectively. The percentage improvement in power varies with benchmarks, varying from 2.7% to 12.7% with most designs around 5-7%. On average, OSFA methodology achieves 6.1% reduction in total power compared to conventional methodology. The run-time in OSFA is about twice that with one scenario for all the designs, and this is intuitive as OSFA needs to compute costs, run STA for 2 scenarios. The run-time for the biggest design ('netcard') of OSFA is around 1.8hr. This is comparable to the run-times in the state-of-the-art gate sizers [4][7] even with one scenario considering only leakage power. More importantly, the run-time in OSFA can be further improved by running on machines with more cores.

Design Space Exploration for More than 2 Operating Conditions: When the number of operating conditions (n) is more than 2, an exhaustive way to perform the design-space exploration is to fix the voltage of one scenario and assign m voltage steps for the rest $n-1$ scenarios, and run OSFA for each case. The complexity of that approach would be $O(m^{n-1})$. To tackle this high computational cost, we propose an alternate way of progressively selecting V_{dd} for each scenario. At the first stage, we consider 2 scenarios, fix V_{dd} in scenario 1 and run OSFA for m voltage steps in 2^{nd} scenario. V_{dd} for the 2^{nd} scenario is selected by taking the minimum power point in the design-space exploration curve. Next, we fix the V_{dd} of 1^{st} and 2^{nd} scenario and run OSFA for m voltage steps in 3^{rd} scenario and so on. To generalize, at the i^{th} stage, we fix V_{dd} of i scenarios, and run OSFA considering $i+1$ scenarios by varying V_{dd} of $(i+1)^{th}$ scenario followed by selecting V_{dd} for the $(i+1)^{th}$ scenario. The overall complexity of this alternative method would be at most $O(m \times (n-1))$. However, this approach might compromise in solution quality to some extent.

Next, we take the design 'pci_bridge32' and create two more operating conditions with clock period twice and 2.5 times that of the fast scenario (with random switching activities). With exhaustive design-space exploration, we get 7.3% and 7.6% improvement in power with $5^{3-1} = 25$ and $5^{4-1} = 125$ OSFA runs for 3 and 4 scenarios respectively. On the contrary, by using the second approach, the power savings reduce slightly to 6.7% and 7.2% with $5 \times (3-1) = 10$ and $5 \times (4-1) = 15$ OSFA runs respectively for 3 and 4 scenarios. These experimental runs demonstrate the following: (i) as the number of operating conditions increases, there is generally more power savings, and (ii) the proposed speed-up technique can reduce the computational cost significantly with little compromise in solution quality.

Table 1: Comparison with conventional method

Design	Cells	Power(W) in Conventional Methodology	Power(W) in OSFA Methodology	% Imprv.
DMA	25,301	0.399	0.375	5.9
pci_bridge32	33,203	0.242	0.226	6.4
des_perf	111,229	2.448	2.381	2.7
vga_lcd	164,891	0.651	0.633	2.7
b19	212,674	0.865	0.755	12.7
leon3mp	649,191	1.528	1.413	7.5
netcard	958,780	2.043	1.951	4.5
Average				6.1

6. CONCLUSION

This work introduces a novel problem formulation of gate-sizing under multiple operating conditions. We present our OSFA algorithms and a design-space exploration methodology to optimize power of any IP-design without affecting the performance at different operating conditions. Compared with conventional methodology, our approach has achieved an average power improvement of 6.1% in industry-strength large-scale benchmarks. We have also proposed a faster yet efficient design-space exploration technique for more than 2 scenarios and demonstrated its effectiveness. We also experimentally observe that the power savings increase with the number of operating conditions. In future, we plan to study the impact of the scenario percentage on our OSFA algorithms. With aggressive technology scaling, the number of operating conditions will further increase, and we believe the OSFA methodology will become more and more relevant in the VLSI industry.

7. ACKNOWLEDGEMENT

This work is supported in part by NSF, SRC and Samsung Semiconductor.

8. REFERENCES

[1] Y. Liu *et al.*, "Multi-scenario buffer insertion in multi-core processor designs," *ISPD*, pp. 15–22, 2008.
[2] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment," *TCAD*, pp. 223–234, 2010.
[3] H. Chou *et al.*, "Fast and effective gate-sizing with multiple-vt assignment using generalized lagrangian relaxation," *ASPDAC*, pp. 381–386, 2005.
[4] J. Hu *et al.*, "Sensitivity-based metaheuristics for accurate discrete gate sizing," *ICCAD*, pp. 233–239, 2012.
[5] M. M. Ozdal *et al.*, "Gate sizing and device technology selection algorithms for high-performance industrial designs," *ICCAD*, 2011.
[6] H. Ren and S. Dutt, "A network-flow based cell sizing algorithm," *IWLS*, 2008.
[7] L. Li *et al.*, "An efficient algorithm for library-based cell-type selection in high-performance low-power designs," *ICCAD*, pp. 226–232, 2012.
[8] V. S. Livramento *et al.*, "A hybrid technique for discrete gate sizing based on lagrangian relaxation," *TODAES*, 2014.
[9] A. Ramalingam *et al.*, "Robust analytical gate delay modeling for low voltage circuits," *ASPDAC*, 2006.
[10] W. N. Li, "Strongly NP-hard discrete gate sizing problem," *TCAD*, pp. 1045–51, 1994.
[11] M. M. Ozdal *et al.*, "The ISPD-2012 discrete cell sizing contest and benchmark suite," *ISPD*, pp. 161–164, 2012.
[12] B. J. Sheu *et al.*, "Berkeley short-channel igfet model for mos transistors," *IEEE Journal of Solid-State Circuits*, pp. 558–566, 1987.
[13] C. P. Chen *et al.*, "Fast and exact simultaneous gate and wire sizing by lagrangian relaxation," *ICCAD*, pp. 617–624, 1998.
[14] `https://www.threadingbuildingblocks.org/`.
[15] `http://ptm.asu.edu/modelcard/HP/45nm_HP.pm`.
[16] `http://www.mentor.com/products/fv/modelsim/`.
[17] Q. Wu *et al.*, "A note on the relationship between signal probability and switching activity," *ASPDAC*, pp. 117–120, 1997.