Discrete Vt Assignment and Gate Sizing Using a Self-Snapping Continuous Formulation

Saumil Shah[1] Ashish Srivastava[1] Dushyant Sharma[2] Dennis Sylvester[1] David Blaauw[1]
Vladimir Zolotov[3]

[1] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, USA
[2]Department of Industrial Operations and Engineering, University of Michigan, Ann Arbor, USA
[3]IBM T.J. Watson Research Center, Yorktown Heights, USA

*Abstract-***This paper presents a novel approach towards the simultaneous Vt-assignment and gate-sizing problem. This inherently discrete problem is formulated as a continuous problem, allowing it to be solved using any of several widely available and highly efficient non-linear optimizers. We prove that, under our formulation, the optimal solution has discrete Vts assigned to almost every gate, thus eliminating the need for a sophisticated snapping heuristic. We show that this technique performs dual-Vt assignment and gate sizing in a very efficient manner. Compared to a sensitivity based method, we achieve average leakage savings of 31% and average total power savings of 7.4% with very efficient runtimes.**

1. Introduction

Due to the increased need for high-performance circuits, low-threshold voltage devices are aggressively used in deep submicron technologies. Subthreshold leakage, being an exponential function of threshold voltage, is becoming an increasingly significant issue. Leakage power is projected to consume approximately half the total power by the 90nm node [1]. These contrasting requirements have made the use of dual-Vt processes inevitable. Critical paths on a circuit are assigned to high-performance, low-threshold voltage devices and non-critical paths are assigned to low-performance, low-leakage, high-threshold voltage devices.

There has been a large amount of work in power optimization using dual-Vt and sizing [2-9]. References [2-4] use sensitivity-based algorithms while [5] employs a Lagrangian relaxation based circuit optimizer. Reference [7] approaches the problem at the transistor level and employs an enumeration based approach along with pruning methods. References [8,9] treat the problem as a continuous optimization problem and heuristically cluster the obtained solution to the discrete domain. A recent method proposed by Chen [10] uses a continuous Vt formulation, where the optimization is performed assuming the availability of a continuous range of threshold voltages. Finally the solution is snapped to one of the physical threshold voltages, requiring a snapping heuristic.

The difficulty of the Dual Vt and sizing problem is that the problem is inherently a Mixed-Integer Non-Linear Program (MINLP) which has a very high complexity level. Sensitivity-based methods are very limited in the design space that they can examine and consequently are inherently heuristic with uncertain quality of solution. On the other hand, the continuous formulations as in [8-10] effectively shift the discrete optimization problem to the snapping phase, where again heuristics are employed and significant discretization error can be incurred.

In this paper, we propose a novel continuous formulation, for which we show that, in the absence of gate width constraints, all gates in the optimal solution automatically snap to one of the discrete Vt values. This eliminates the need for a sophisticated heuristic for Vt discretization and allows the use of a wide range of powerful industrial non-linear optimizers to improve the solution quality and achieve runtime efficiency. In our formulation, each gate is modeled as a parallel combination of high and low Vt gates. The effective Vt of this mixed gate lies between the two extreme threshold voltages [11-12] and depends on the fraction of total width assigned to each Vt portion. The problem is thus formulated in a continuous manner where the two widths are separate optimization variables. We then provide a rigorous proof showing that the optimal solution to this problem formulation, in the absence of gate width constraints, has only one non-zero width value for every gate. Clearly, this approach allows us to perform simultaneous dual-Vt assignment and transistor sizing using a seamless non-linear optimization process and avoids the difficulty present in previous MINLP approaches.

We then show that, in the presence of constraints on the gate widths, a limited number of gates in the optimal solution can have non-snapped Vt values where both the high-Vt and low-Vt portion of the gate have non-zero width. We derive rigorous bounds on the number of occurrences of such non-snapped gates and show that this number is extremely small. In practice, we found that the number of non-snapped gates was less than 2% of the total number of gates. Due to its small number, this small remnant of non-snapped gates can be trivially snapped with a simple heuristic. We show that the impact of this snapping on optimality of the results was much less than one percent.

We also show that the properties of our formulation hold even in the general case of multiple threshold voltages.

However, it is in the common case of a dual Vt process, where the discretization error for the previous approaches is large, that our method would prove to have the greatest advantage. We implement the proposed approach and show that the proposed method achieves average leakage savings of up to 31% and average total power savings of 7.35% over a sensitivity-based approach.

The rest of the paper is organized as follows. Section 2 introduces the traditional single-Vt convex sizing problem and the formulation of the dual-Vt assignment problem in continuous form. Section 3 provides a rigorous proof of the snapping phenomenon explained above. In this section we also extend the analysis to include the practical constraint of fixed-width input drivers. In Section 4 we describe the details of the implementation and our experimental setup. Section 5 gives a detailed discussion of the results and Section 6 concludes the paper.

2. Problem Formulation

The single-Vt, gate sizing problem is traditionally formulated as given below:

$$\text{Minimize: } \sum_{i\in G} P_i W_i$$

$$\text{Subject to: } \sum_{i\in p} D_i \le A_0 \qquad \forall p \in P \qquad (1)$$

$$L_i \le W_i \le U_i \qquad i = 1,...,n.$$

P is the set of all paths from the primary inputs to the outputs, G is the set of all gates, P_i is the power per unit width (static + dynamic) of gate i, W_i and D_i are the width and delay of gate i, respectively. A_0 is the constraint on total circuit delay and the L_i's and U_i's are the bounds on gate size. Clearly, the number of possible paths is exponential in the number of gates n, making this formulation impractical for efficient optimization algorithms. We use the standard technique of partitioning path delay constraints into nodal constraints [13]. We assign a variable a_i to each node i in the circuit, representing the arrival time at node i. Now, the primal problem can be formulated as given below:

$$\text{Minimize: } \sum_{i\in G} P_i W_i$$

Subject to:

$$a_j \le A_0 \qquad j \in \{outputs\}$$

$$a_j + D_i \le a_i \qquad i \in (\{1,...,n\} - \{inputs\}) \qquad (2)$$

$$D_i \le a_i \qquad i \in \{inputs\}$$

$$L_i \le W_i \le U_i \qquad i = 1,...,n.$$

This is the standard convex sizing problem, where a circuit is sized for minimum power while meeting a fixed delay constraint. When this problem is combined with the dual-Vt assignment problem, it loses its convexity and is, in fact, a MINLP. Mixed-Integer problems cannot be solved efficiently in polynomial time and require heuristic solutions. We avoid dealing with the intractability of MINLP's by extending this problem in the *continuous* domain to the simultaneous sizing and dual-Vt assignment problem.

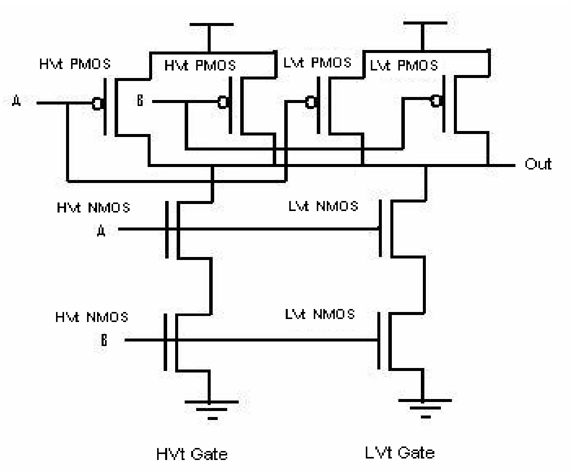


Fig. 1. NAND Gate represented as a parallel combination of HVt and LVt gates

2.1 Continuous Formulation of Dual Vt Problem

To transform the discrete dual-Vt assignment problem into a continuous one, we propose the following formulation. Each gate is considered to be a parallel combination of a high Vt portion and a low Vt portion. Figure 1 shows the equivalent circuit representation of a mixed gate. The effective drive strength and power consumption of the mixed gate are intermediate between high and low Vt parameters. Effectively, this gate can be considered to have a threshold voltage somewhere between the two extremes [11,12].

The equivalent resistance of the mixed gate can be written as

$$R_{eff} = R_{lgate} \parallel R_{hgate}$$

where $R_{lgate} = \frac{R_l}{W_l}$ and $R_{gate} = \frac{R_l}{W_l}$ are the total gate resistances of the Low Vt (LVt) fraction and High Vt (HVt) fraction respectively and R_l and R_h are the resistances per unit width. The effective resistance is, therefore,

$$R_{eff} = \frac{R_l R_h}{R_l W_h + R_h W_l} \qquad (3)$$

For the purpose of this work, we use a delay model where the drive strength of a gate is linearly dependent on its size. The model also allows for capacitive self-loading, where the load capacitance of a gate is a function of its own width along with the widths of its fanouts. This is similar to the models that form the basis of logical effort theory, and have been successfully used in logic synthesis. Under this model, the delay of a gate can be written as

$$D = RC_l \qquad (4)$$

$$C_l = C_{Load} + K_{SL}W$$

Here K_{SL} is a constant that models the contribution of a gate's intrinsic capacitance to its own load.

The delay of the mixed gate is

$$D = R_{eff}C_l = \frac{R_l R_h}{R_l W_h + R_h W_l} C_l \qquad (5)$$

$$C_l = C_{Load} + K_{SL}(W_l + W_h)$$

The total power of every gate can now be written as

$$P_{gate} = P_l W_l + P_h W_h \qquad (6)$$

where P_l and P_h are the power per unit width of the LVt and HVt fraction of the gate respectively. These constants include both dynamic and static power, and are dependent on the load and switching activity of a gate.

Now the problem can be rewritten as

Minimize: $\sum_{i \in G} P_{l,i}W_{l,i} + P_{h,i}W_{h,i}$

Subject to:

$$\begin{array}{ll} a_j \le A_0 & j \in \{outputs\} \\ a_j + D_i \le a_i & i \in (\{1,...,n\} - \{inputs\}) \\ & j \in \{input(i)\} \\ D_i \le a_i & i \in \{inputs\} \\ 0 \le W_{l,i} & i = 1,...,n. \qquad (7) \\ 0 \le W_{h,i} & i = 1,...,n. \\ L_i \le W_{l,i} + W_{h,i} \le U_i & i = 1,...,n. \end{array}$$

The expression for the gate delay appearing in the constraints is

$$D_i = \frac{R_{l,i}R_{h,i}}{R_{l,i}W_{h,i} + R_{h,i}W_{l,i}} C_{l,i}$$

Also, for the primary outputs

$$C_{l,i} = C_L + K_{SL}(W_{l,i} + W_{h,i}) \qquad (8)$$

For all other gates

$$C_{l,i} = \sum_{j \in fo(i)} (C_{inp,j}(W_{l,j} + W_{h,j})) + K_{SL}(W_{l,i} + W_{h,i}) \qquad (9)$$

3. Proof of Discrete-Vt (Snapped) Optimal Solution

In this section, we prove that, ignoring the constraints on device size, the optimal solution has every gate in either the fully high or fully low-Vt configuration.

We prepare a background for the proof by conceptually separating the problem into two phases. In the first phase, which we call the D-phase, we obtain a vector of all gate delays. Carrying this fixed delay vector to the second phase or the W phase, we find the sizing solution for that particular delay vector which gives minimum total power. We now prove that the optimal sizing solution for any arbitrary delay vector has every gate snapped to either LVt or HVt. It is easy to see that this condition holds for the optimal solution of the complete problem too, since the optimal delay vector is simply a special case of an arbitrary combination of delay assignments. We clarify that, although the separation of the problem into two phases is conceptual and does not reflect the actual optimization procedure, the validity of the argument is independent of the formulation used by the optimizer.

The W-phase of the problem (7) can be written as

Minimize: $\sum_{i \in G} P_{l,i}W_{l,i} + P_{h,i}W_{h,i}$

Subject to:

$$R_{l,i}W_{h,i} + R_{h,i}W_{l,i} = \frac{R_{l,i}R_{h,i}\left(\sum_{j \in fanout(i)} (C_{inp,j}(W_{l,j} + W_{h,j})) + K_{SL}(W_{l,i} + W_{h,i})\right)}{D_i} \quad i = 1,...,n.$$

$$\begin{array}{ll} W_{l,i} \ge 0 & i = 1,...,n. \qquad (10) \\ W_{h,i} \ge 0 & i = 1,...,n. \end{array}$$

In this sub-problem, the D_i's are treated as constants carried over from the D-phase. It is interesting to note that the objective function and all constraints are linear; therefore this is a Linear Programming Problem (LPP). For clarity, we write the problem in the canonical LPP form as shown in (11).

Minimize: $P^T(W_l, W_h)$

Subject to:

$$A^T(W_l, W_h) = B \qquad (11)$$

$$(W_l, W_h) >= 0$$

Here P^T is the vector of P_l and P_h values, (W_l, W_h) is the vector of LVt and HVt gate widths, A is the n×n matrix of the coefficients of the equality constraints, and B is the vector of the constants appearing in the equality constraints. With this background, we present a formal proof of the snapping phenomenon.

Theorem 1: The optimal solution to the problem in (11) has the property that the width of every gate has exactly one non-zero component.

i.e. $\forall i$, $W_{l,i} = 0$ or $W_{h,i} = 0$

Proof: We note that, in the non-degenerate case, all equality constraints are linearly independent. Therefore A is a full-rank matrix with rank n.

The problem stated in (11) is a LPP with 2n variables, n equality constraints, and 2n non-negativity constraints.

We know that the power cannot be negative, hence the solution is bounded and is always attained. Therefore, this LPP must have a finite solution. From the Theory of Linear Programming [15] we know that if this LPP has a solution, it is a basic feasible solution, which has (2n-n) basic variables and (2n-n) non-basic variables. The basic variables are free to have any value and the non-basic variables are constrained to be at their lower bound (which is zero).

Thus, it is clear that in the optimal solution, n variables will have value zero and n variables will be non-zero. It is obvious that for any gate it is impossible for both W_l and W_h to have zero value, because the delay of that gate would be infinitely large. Hence, it is clear that each gate has one zero W component and one non-zero component. Therefore each gate is snapped to either high or low Vt.□

3.1 Extension to circuits with fixed-width input driver constraints

Having proved the snapping phenomenon under the proposed model, we now add a practical constraint. In practice, combinational circuits are usually driven by sequential elements, the sizes of which are not included in the combinational optimization procedure. The delays of these sequential elements, however, must be considered while computing the total circuit delay since their delays are affected by the sizes of the gates driven by them. Ignoring this constraint allows the primary input gates to be sized up indefinitely without incurring any delay penalty, which could lead to an impractical solution. To model this constraint, we introduce fixed-width input drivers feeding all the primary inputs. This concept is clearly explained in Figure 3, which illustrates a circuit with fixed-width input drivers. In comparison, the circuit shown in Figure 2 has no drivers and therefore upsizing gate 1 does not incur any delay penalty. All previous approaches referenced in Section 1 implicitly make the simplifying assumption of Figure 2 without providing further analysis. We remove this assumption and prove that the significance of our results is not diminished. Although, under this constraint we cannot conclusively prove that every gate will snap, a simple extension of the argument made above can prove that only a very small proportion of gates (if any) will be non-snapped.

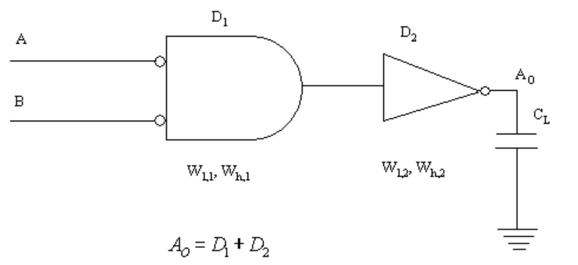
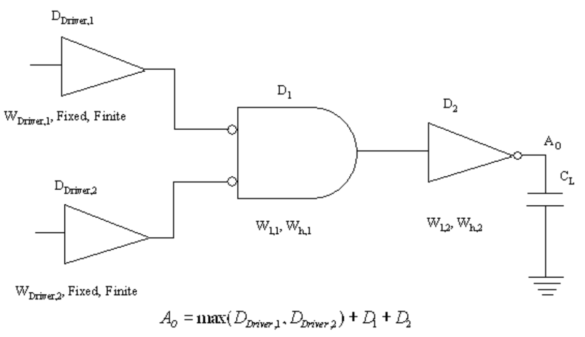


Fig. 2. Circuit in absence of fixed-width input drivers



$D_{Driver,1}$
$W_{Driver,1}$, Fixed, Finite
D_1
D_2
A_0
C_L
$D_{Driver,2}$
$W_{l,1}, W_{h,1}$
$W_{l,2}, W_{h,2}$
$W_{Driver,2}$, Fixed, Finite
$A_0 = \max(D_{Driver,1}, D_{Driver,2}) + D_1 + D_2$

Fig. 3. Circuit with fixed-width input drivers

Let us consider a circuit with m primary inputs, and therefore, m input drivers. We proceed in exactly the same manner as before, separating the problem into the two phases. In the W phase, we consider the delays of all gates (including the input drivers) to be fixed. The problem formulation remains the same as (11), except for the addition of m constraints imposed by the delay equations of the input drivers. Clearly, this new problem is also an LPP. The rank of the matrix A now changes to n+m. We, therefore, have a canonical form LPP with 2n variables, n+m constraints and 2n non-negativity constraints. Once again, from the Theory of Linear Programming, there are 2n-(n+m) non-basic variables and n+m basic variables. Therefore, in a non-degenerate solution, n+m variables are positive and n-m variables have value zero. This implies that a total of m gates could possibly be non-snapped (have both their W's non-zero), where m is the number of primary inputs. Hence, the total number of non-snapped gates is bounded by the number of primary inputs, which is usually a small proportion of the number of gates in the circuit. We also note that, although the number of primary inputs is an upper bound on the number of non-snapped gates, in practice the number is found to be much smaller. Over most circuits, this was found to be zero, and, as shown in Fig. 6, always proved to be a very small fraction of the total number of gates. In Section 4, we explain how this small proportion of gates can be handled heuristically without much difficulty.

3.2 Non-snapped gates due to maximum and minimum sizing constraints

When upper and lower bounds are placed on total device size, as in formulation (7), we have n additional constraints in the LPP. The snapping property does not explicitly hold under Theorem 1. However, the non-snapped gates are only those that are at their extreme values in the optimal solution. Clearly, in optimized digital circuits, maximum size gates are very rare. It is often more profitable to assign a gate to the LVt configuration than to size it beyond a certain bound. On the other hand, minimum size gates rarely lie on critical paths. Since they are not critical, small changes to their delay do not affect total circuit delay. Also, their contribution to power is relatively small. Therefore, snapping their Vt heuristically does not incur a significant delay or power penalty.

3.3 Generalization of proof to a k-Vt (k>2) process

The proof in Section 3 can easily be extended to a process where there are more than two allowable threshold voltages. Each gate can be considered a parallel combination of n gates of different threshold voltages. The delay of gate i can now be written as

$$D_i = \frac{\prod_{j=1}^{k} R_{j,i}}{\sum_{m=1}^{k} (\prod_{\substack{j=1 \\ j \neq m}}^{k} R_{j,i}) W_{m,i}} C_{l,i} \tag{12}$$

Where $R_{j,i}$ is the resistance of the jth Vt fraction of the ith gate and $W_{k,i}$ is the width of the kth Vt fraction of the ith gate. Multiplying on both sides of the equation by the denominator, once again we get a linear equation. Formulating this in exactly the same manner as the dual-Vt problem, we obtain a LPP with k×n variables, n equality constraints and k×n non-negatvity constraints. As stated above, the optimal solution has (k-1)×n non-basic variables. Therefore, only n variables can be non-zero. As stated before, each non-zero variable has to be a width fraction from a different gate. If it were not so, we would have gates with zero total width. Clearly, if we use this formulation to perform multi-Vt optimization, we end up getting a solution where one particular Vt is assigned to each gate. We have thus shown that we can use our method to successfully perform multiple-Vt assignment.

4. Implementation Details and Experimental Setup

We use the method explained above to perform dual-Vt assignment on large-scale combinational circuits. We hypothetically represent every gate as a combination of high and low Vt gates and formulate the problem as shown above. We first modify the problem to improve convergence properties of the optimizer by applying the exponential transform to all variables. The final variables used in the optimization problem are the logarithmic values of the variables in the equations given above. This choice of variables and variable transformations was made by conducting experiments with different formulations. It is intuitive that the exponential transformation should give best convergence properties as it is the commonly used transformation for convexification of geometric programming problems. We express the model using the mathematical programming language AMPL [16]. The AMPL interface can be used to invoke a wide range of commercial non-linear solvers. For this work, our solver of choice is MINOS [17] due to its stability and efficiency.

The solution returned by the solver has very few gates that are non-snapped due to the sizing constraints (usually the lower bound) and even fewer gates that are non-snapped because of the fixed input drivers. We propose two heuristics to arrive at a solution with an entirely discrete Vt assignment.

Heuristic 1

In the first approach, we round the non-snapped gates to closer of the two threshold voltages. We thus fix the Vt of these gates by forcing one of the two W components to be zero and rerun the optimization. The new solution obtained in this way has certain previously-snapped gates in the non-snapped configuration. However, the number of non-snapped gates obtained in an iteration is always lower than in the previous iteration. We repeat this procedure iteratively until a fully snapped solution is reached. The pseudocode for this heuristic is given in Fig. 4. An interesting property is observed in the solution obtained at the end of this iterative procedure. Even in large circuits, only 1 or 2 gates are found to have differing Vt values in the final solution compared to the first pass. This observation leads us to our second heuristic, which is far simpler and achieves better runtime negligible penalty in the quality of results.

Heuristic 2

After the initial optimization run, we fix the Vt of every gate, rounding non-snapped gates to the closest Vt value. Subsequently we perform a sizing run to meet timing. A comparison of the total power for the two heuristics showed the solutions to be within 0.1% of each other. We, therefore, use the second heuristic to compare our results with previously proposed approaches.

```
1.  Algorithm : CONT_VT1
2.  write_AMPL_input();
3.  iterate:
4.  run_MINOS();
5.  parse_out_file();
6.   if non-snapped gates = 0
7.      end;
8.   else
9.      fix_non_snapped_gates();
10. goto iterate;
```

Fig. 4. Pseudocode for continuous sizing and Vt assignment using Heuristic 1

```
1.  Algorithm : CONT_VT2
2 .  write_AMPL_input();
3.   run_MINOS();
4.   parse_out_file();
5.   fix_non_snapped_gates();
6.   run_MINOS();
7 .  end;
```

Fig. 5. Pseudocode for continuous sizing and Vt assignment using Heuristic 2

At this point, we note that this problem is not convex; therefore it is not possible to arrive at a globally optimal solution in polynomial time. However, our approach returns a local minimum, and, with a suitable choice of initial solution, is guaranteed to perform better than a sensitivity-based algorithm.

We use an industrial 0.13μm technology with primary input activity factors adjusted to maintain a 70/30 ratio of dynamic to static power across all circuits. The Vts are such that the ratio of the resistances is 1.15/1 and the leakage ratio is 10/1. To compare the proposed solution to the sensitivity-based algorithm (SBA) in [2], we take the solution returned by the optimizer and discretize the sizes by snapping them to the closest library cell. Although, this does alter the delay constraint to some extent, the timing penalty is fairly small (~2-3%). We use this new delay value as the constraint for SBA and optimize for total power. We compare the two algorithms in terms of achieved objective function value (total power) as well as in terms of runtime.

5. Results

This section gives detailed results of our experiments on circuits from the ISCAS [18] and MCNC [19] benchmark suites. Figure 2 shows the snapping properties of two circuits – c5315 (1750 gates) and c7552 (1994 gates) to illustrate the proportion of non-snapped gates as a function of the timing backoff. The backoff is defined with respect to the best possible delay achievable by the optimizer for a particular circuit. It is clearly seen that the number of non-snapped gates is very small to start with and decreases as the delay constraint is relaxed. Also, a majority of the non-snapped gates are those that have reached the constraint on their total size, which are easy to handle as stated earlier. The number of non-snapped gates due to fixed-width input drivers is zero over most timing constraints and extremely small otherwise. Table 1 shows a comparison of results obtained by the two heuristics and the power compared to the initial non-snapped solution. The number of iterations required for the convergence of Heuristic 1 is also shown in the table. Heuristic 2, by definition, always requires exactly two iterations. For brevity we include only results for the larger circuits in Table 1. Table 2 compares the total power obtained by our algorithm to SBA-based optimization. To compare the results we heuristically snap the gate sizes to a fine- grained library, where each gate is 1.2X larger than the previous gate of its type. We observe that circuits optimized by our method have considerably lower leakage than those optimized by SBA. We also show gains in total power, achieved by a trade-off between dynamic power and static power. Clearly, circuits that have a higher proportion of leakage power initially will exhibit more substantial total power improvements. The results show that the optimizer allows a small dynamic power penalty in order to reduce leakage and hence total power. This observation clearly illustrates

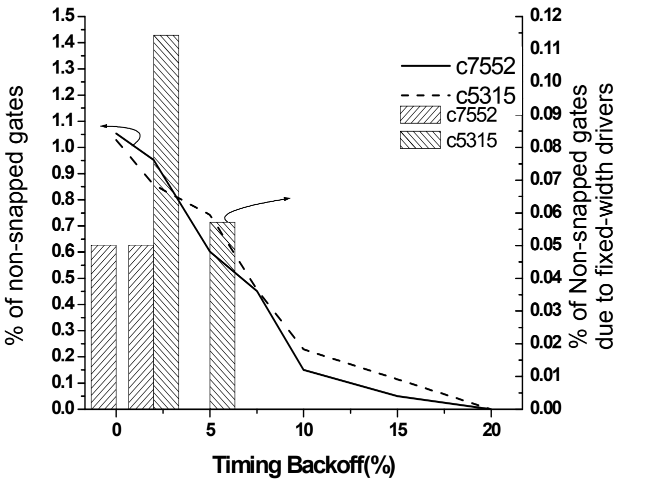


Fig. 6. Number of non-snapped gates as a function of circuit delay for c5315. The lines shows the total number of non-snapped gates while the histogram shows only those which are not at a sizing bound.

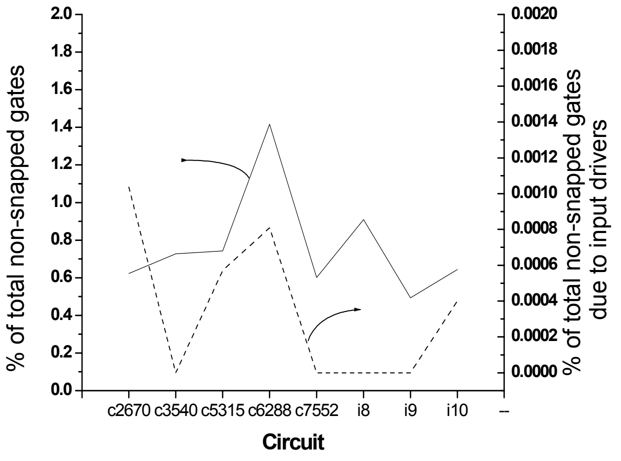


Fig. 7. Percentage of non-snapped gates at 5% timing backoff for large circuits

Table 1. Comparisons of proposed heuristics for large circuits at 2% timing backoff. Total power is normalized to initial non-snapped solution

Circuit	Heuristic 1		Heuristic 2	
	Iterations	Total Power	Iterations	Total Power
c3540	7	1.001	2	1.002
c5315	3	1.002	2	1.002
C6288	8	1.006	2	1.007
C7552	4	1.004	2	1.004
i8	3	1.002	2	1.002
i9	2	1.002	2	1.002
i10	2	1.003	2	1.003

the advantage of using an algorithm that handles gate sizing and Vt-assignment as a unified problem rather than independently. The table also shows runtime comparisons.

For most circuits the runtime of one iteration of the optimizer is better than the runtime exhibited by SBA. However, the complete procedure involves two iterations and the proposed approach therefore suffers some runtime penalty.

We also observe that our formulation allows us to meet much tighter timing constraints (up to 10%) than can be met by SBA. We attribute this to the fact that the sizing problem is convex and can be handled very efficiently by the non-linear optimizer, especially if the initial solution passed to the optimizer is an all-LVt design, The sensitivity-based heuristic, on the other hand, will not necessarily converge to the optimal solution, even if the problem is convex, and, therefore, suffers from sub-optimality.

6. Conclusions

We present a novel technique to solve the dual-Vt assignment problem. We formulate the problem in a continuous manner and show that the optimal solution has a very large proportion of the total gates already assigned to one of the two threshold voltages. We snap the few remaining gates heuristically. The main contribution of this work is providing a formulation that can be solved efficiently by a general purpose non-linear optimizer to obtain a discrete dual-Vt solution without the added penalty of heuristic Vt discretization. The optimization procedure is thus more effective than sensitivity-based or separate sizing and Vt-assignment methods proposed previously. Circuits optimized by the proposed technique have 31% lower leakage and 7.4% lower total power on average than circuits optimized using a sensitivity-based algorithm.

Acknowledgements

This work was supported by funding from MARCO/DARPA GSRC.

References

[1] S. Narendra *et al*., "Leakage Issues in IC Design: Trends, Estimation and Avoidance", *Tutorial, ICCAD,* 2003.

[2] S. Sirichotiyakul *et al*., "Duet: An Accurate Leakage Estimation and Optimization Tool for Dual-Vt Circuits", *IEEE Transactions on VLSI Systems*, pp. 79-90, April 2002.

[3] P. Pant, R. Roy, and A. Chatterjee. "Dual-threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits," *IEEE Trans. on VLSI Systems,* pp.390-394, 2001.

[4] L. Wei *et al*., "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS circuits," *Proc. Design Automation Conference,* pp. 489-494.1998.

[5] T. Karnik, *et al*., "Total Power Optimization by Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessors," *Proc. Design Automation Conference,* pp.486-491, 2002.

[6] D. Nguyen, *et al.,* "Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization," *Proc. International Symposium on Low-Power Electronics Design*, pp. 158-163, 2003.

[7] M. Ketkar, *et al*., Convex Delay Models for Transistor Sizing," *Proc. Design Automation Conference,* pp. 655-660, 2000.

[8] A. Srivastava, *et. al.* "Simultaneous Vt Selection and Assignment for Leakage Optimization," *Proc. International Symposium on Low-Power Electronics Design*, pp. 146-151, 2003.

[9] V. Sundarajan and K. Parhi, "Low Power Synthesis of Dual Threshold Voltage CMOS VLSI circuits," *Proc. International Symposium on Low-Power Electronics Design*, pp. 139-144, 1999.

[10] C. Chen *et. al*. "Fast and Effective Gate-Sizing with Multiple-Vt Assignment using Generalized Lagrangian Relaxation", *Proc. Asia South Pacific - Design Automation Conference,* pp. 381-386. 2005.

[11] K. Agarwal, *et. al.* "Achieving Continuous V_t performance in a dual-Vt process", *Proc. Asia South Pacific – Design Automation Conference,* pp. 393-398, 2005.

[12] S. Shah *et. al.* "A New Threshold Voltage Assignment Scheme for Runtime Leakage Reduction in On-Chip Repeaters", *Proc. Intl. Conf. on Computer Design,* pp. 138-143, 2004.

[13] C. Chen *et. al*., "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," *IEEE Transactions on CAD,* pp. 1014-1025, July 1999.

[14] H. B. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI," Addison-Wesley, 1990.

[15] M.S. Bazaraa *et. al.* "Linear Programming and Network Flows," *3rd Edition,* Wiley, 2005.

[16] R. Fourer, D. M. Gay and B. W. Kernighan, "A Modeling Language for Mathematical Programming," *Management Science, Vol. 36,* pp. 519-554, 1990.

[17] B. A. Murtagh and M. A. Saunders, "MINOS 5.4 User's Guide, Report SOL 83-20R," *Systems Optimization Laboratory, Stanford University,* December 1983 (revised February 1995).

[18] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," Proc. ISCAS, pp. 695-698, May 1989.

[19] http://www.cbl.ncsu.edu

Table 2. Comparison of total power consumption (normalized to total SBA power) between circuits optimized with continuous method and circuits optimized with SBA

Ckt	Timing Backoff	SBA		Continuous Formulation			% Improvement		Runtime (s)	
		Static	Dyn,	Static	Dyn.	Total	Static	Total	SBA	Cont.
c432	2%	0.34	0.66	0.13	0.63	0.75	62.35	24.69	4	3
	5%	0.21	0.79	0.12	0.68	0.79	45.74	20.72	3	2
	10%	0.16	0.84	0.10	0.77	0.87	39.74	13.01	3	2
c1908	2%	0.19	0.81	0.10	0.84	0.93	48.57	6.76	21	11
	5%	0.16	0.84	0.09	0.86	0.95	44.71	4.94	25	8
	10%	0.15	0.85	0.08	0.87	0.95	44.74	5.01	24	8
c2670	2%	0.26	0.74	0.19	0.74	0.93	29.13	7.28	58	37
	5%	0.26	0.74	0.17	0.74	0.91	36.36	9.11	46	35
	10%	0.25	0.75	0.16	0.77	0.93	34.29	6.56	47	26
c3540	2%	0.26	0.74	0.16	0.78	0.94	38.14	6.46	28	51
	5%	0.24	0.76	0.14	0.81	0.94	41.32	5.54	26	47
	10%	0.23	0.77	0.13	0.80	0.93	42.76	6.97	23	40
c5315	2%	0.22	0.78	0.15	0.80	0.95	30.53	5.11	52	133
	5%	0.21	0.79	0.16	0.80	0.96	23.60	3.72	50	119
	10%	0.20	0.80	0.14	0.82	0.96	28.40	3.62	62	125
c6288	2%	0.35	0.65	0.26	0.65	0.91	24.69	9.04	136	443
	5%	0.31	0.69	0.24	0.69	0.93	22.69	7.22	140	517
	10%	0.24	0.76	0.19	0.76	0.96	18.66	4.34	130	471
c7552	2%	0.31	0.69	0.24	0.68	0.91	23.93	8.87	94	171
	5%	0.30	0.70	0.22	0.71	0.93	26.57	7.00	80	143
	10%	0.28	0.72	0.24	0.72	0.96	16.57	4.08	81	131
i2	2%	0.32	0.68	0.22	0.69	0.91	31.88	8.88	2	2
	5%	0.29	0.71	0.20	0.71	0.91	29.82	8.59	2	2
	10%	0.25	0.75	0.17	0.75	0.92	29.55	7.82	1	2
i3	2%	0.21	0.79	0.11	0.79	0.89	50.00	10.55	2	3
	5%	0.19	0.81	0.08	0.81	0.90	55.56	10.05	1	2
	10%	0.16	0.84	0.07	0.83	0.91	53.57	9.20	1	2
i4	2%	0.20	0.80	0.10	0.81	0.91	50.70	9.14	3	3
	5%	0.17	0.83	0.07	0.83	0.90	61.02	9.97	3	2
	10%	0.14	0.86	0.07	0.87	0.93	52.27	6.71	2	2
i5	2%	0.23	0.77	0.21	0.74	0.95	6.78	4.69	5	7
	5%	0.21	0.79	0.20	0.77	0.97	5.88	2.51	4	3
	10%	0.20	0.80	0.17	0.80	0.97	13.33	3.04	4	4
i6	2%	0.26	0.74	0.20	0.74	0.95	22.92	5.46	6	6
	5%	0.22	0.78	0.20	0.75	0.95	10.26	5.08	6	5
	10%	0.23	0.77	0.21	0.74	0.95	10.26	4.79	6	5
i7	2%	0.22	0.78	0.12	0.79	0.90	46.02	9.73	12	21
	5%	0.19	0.81	0.12	0.81	0.93	38.95	7.43	9	20
	10%	0.18	0.82	0.12	0.81	0.93	33.71	7.22	10	17
i8	2%	0.24	0.76	0.19	0.75	0.94	21.57	5.87	24	35
	5%	0.24	0.76	0.18	0.75	0.94	24.00	6.28	19	33
	10%	0.22	0.78	0.19	0.77	0.97	12.64	3.30	23	37
i9	2%	0.20	0.80	0.16	0.77	0.94	17.65	6.47	9	21
	5%	0.19	0.81	0.17	0.78	0.96	11.29	4.40	8	23
	10%	0.20	0.80	0.17	0.79	0.96	13.11	3.55	8	19
i10	2%	0.31	0.69	0.23	0.69	0.92	24.58	7.69	287	373
	5%	0.32	0.68	0.24	0.68	0.93	22.51	7.04	276	389
	10%	0.32	0.68	0.24	0.68	0.92	25.23	7.89	293	351
Avg.Imp							31.22%	7.36%		