A Scalable Algorithmic Framework for Row-Based Power-Gating

A.Sathanur
Politecnico di Torino

A.Pullini
Politecnico di Torino

L.Benini
Università di Bologna

A.Macii
Politecnico di Torino

E.Macii
Politecnico di Torino

M.Poncino
Politecnico di Torino

ABSTRACT
Leakage power is a serious concern in nanometer CMOS technologies. In this paper we focus on leakage reduction through automatic insertion of sleep transistors for power gating in standard cell based designs. In particular, we propose clustering algorithms for row-based power-gating methodology which is based on using rows of the layout as the granularity for clustering. Our clustering methodology does timing and area constraint driven power-gating in contrast to only timing driven power-gating as proposed in the previous works. We present two distinct clustering algorithms with different accuracy-efficiency trade-off. An optimal one, which exploits a 0-1 or Binary Integer Programming approach, and a heuristic one, which resorts to an implicit enumeration of the layout rows. Results show that, for all the benchmarks, the leakage power savings, as compared to previous techniques, are more than 75% when we have the same timing constraints but half sleep transistor area and at least 60% when area constraint is set at one fourth. We also show that we can perform clustering with no speed degradation and achieve maximum leakage power savings up-to 83%.

1. Introduction
Leakage power is a major concern in sub-90nm CMOS technology and numerous design techniques have been proposed to reduce standby leakage in digital circuits (see [1, 2] for exhaustive surveys on the topic). Out of this extensive range of solutions, power-gating (a.k.a. sleep transistor insertion or MTCMOS) has proven to be a very effective approach to reduce standby leakage, while keeping high speed in the active mode. In its basic form, power gating is based on the addition of devices (the sleep transistors) in series with the pull-up and/or the pull-down network of a logic gate, and turning them off when the circuit is idle, thereby decreasing the leakage power component due to sub-threshold currents. Gating the pull-down network only is generally preferred because of the large on-resistance of PMOS switches. When a NMOS sleep transistor is used, a sleep signal controls its active/standby mode. In the standby mode, the sleep transistor is off, thus disconnecting its insertion point, called virtual ground, from the physical ground. In active mode the gated device operates normally, but it incurs a delay degradation due to the series resistance of the sleep transistor.

Effective use of power-gating requires a proper sizing of the sleep transistor, since that affects the performance of all the gates connected to it. While a small transistor unacceptably slows down the circuit in active mode due to its high resistance, a large one implies a significant overhead in area and a non-negligible energy (i.e., power and delay) for ON–OFF transitions. An additional challenge in sizing the sleep transistor is to accurately estimate the maximum current drawn by the power-gated design since it directly impacts the performance of the gates in the design. Another fundamental problem to be addressed in power gating concerns the granularity of the blocks to which gating is applied. Granularity may range from individual cells to large chip sub-units. Again, the choice spans a trade-off between effectiveness and efficiency: fine-grain or cell-level gating requires one sleep transistor for each cell, but has maximum optimization potential; on the other extreme, coarse-grain gating has smaller overhead as compared to the fine grain approach but also has smaller potential for optimization.

Clustered power gating, in which a set of cells is identified and gated by the same sleep transistor, has emerged as a very promising solution to determine an intermediate granularity between the above two extremes. However, clustered sleep transistor insertion still lacks efficient clustering algorithms that take into account physical design details, as well as full tool support in industrial EDA frameworks.

A recent work has proposed a novel power gating methodology for standard-cell based designs known as row-based power-gating in which the granularity is fixed (an entire row of the layout), and sleep transistors are placed in dedicated rows [12]. This approach still falls in the family of clustered approaches, but now a cluster consists of a set of rows of the layout. Such a scheme improves the scalability of the algorithm drastically as compared to cell-level solutions. Furthermore, using layout rows as atomic clustering objects greatly simplifies the physical-level management of virtual ground distribution, which is the major practical obstacle in clustering approaches that work on a cell-by-cell basis. The authors show that this methodology incurs least perturbation of the original layout and as a consequence is extremely layout friendly thus helps in automating the insertion of sleep transistors and to achieve faster design closure.

This paper adopts the row-based paradigm of [12] and improves it by addressing the main issues that remained open in that work. More precisely, we provide the following contributions:

- An optimal as well as a heuristic clustering algorithms that solve the problem of leakage minimization under user-specified timing constraints;
- A demonstration of the claimed scalability of the row-based paradigm by providing results for large designs.
- An assessment of the potential of the row-based paradigm as an exploration environment, by presenting exploration results of the complex leakage/performance/area trade-off.

The paper is organized as follows. Section 2 provides an overview of the literature on power gating, and in particular on clustered
approaches. Section 3 provides a detailed insight into the exploration potential offered by row-based power gating. Section 4 describes the proposed clustering methodology and related algorithms, whose effectiveness is benchmarked in Section 5. Finally, Section 6 concludes the work.

2. Previous Work

The first work on sleep transistor insertion for leakage power reduction appeared in [3] where the authors propose the use of sleep transistors for power management. But the authors do not address the issues of sizing and clustering of sleep transistors in this article. In [5], the authors propose a solution for sizing sleep transistors and implicitly address the issue of clustering. They show that grouping the gates which have mutually exclusive discharge current pattern can result in very optimal sleep transistor sizes. But the authors fail to provide a complete and a comprehensive methodology. Later in [6], the authors present several heuristics for efficient gate clustering in CMOS circuits using bin-packing (BP) and set-partitioning (SP) techniques. First they partition the design into sub-clusters with gates discharging current not more than a specified maximum value. Then they find minimum number of bins or groups of sub-clusters where each group has one sleep transistor assigned to it with a specified maximum current sinking capability. They also perform clustering taking into account the proximity of gates in the layout. But in this work, the authors only consider the timing constraints while sizing the sleep transistor and hence their methodology do not constrain the sleep transistor area. In [7], the authors show that sizing based on time varying discharge current pattern instead of just the peak discharge current can lead to lower sleep transistor sizes as compared to [6]. However one major drawback of this methodology is that there is no control on the virtual ground voltage while sizing and a large value on virtual ground voltage can lead to signal integrity problems.

In [9], the authors propose to consider not only the topology of a circuit but also the functional information about the circuit for clustering leading to lower sleep transistor sizes. But a major disadvantage of this solution is that the proximity of the gates in the layout while building the clusters is not considered. Several other methodologies for sleep transistor insertion considering detailed layout information are also proposed. In [8], the authors propose a new methodology known as distributed sleep transistor network (DSTN) where multiple clusters are connected by a virtual ground mesh. They show that this methodology reduces the total size of the sleep transistor due to discharge current balancing. In [10] the authors propose a timing driven power-gating methodology for DSTN and provide efficient algorithms for sizing the sleep transistor. Both these methodologies which work on DSTN requires major changes in the power distribution policy since it requires routing of virtual ground mesh connecting a number of clusters thus limiting its use to a few clusters.

But none of the above proposed methodologies consider sleep transistor area as a design constraint while clustering and sizing the sleep transistor. This is major fall back since very huge sleep transistors can easily lead to a multitude of problems as explained in the previous section.

3. Timing- and Area-driven Power-Gating

Using layout rows as atomic gating elements is by itself just another choice of granularity, and does not really imply a new methodology. What makes this approach truly innovative is the possibility of selectively applying gating only to portions (in our case, a subset of the rows) of a design; the selection of this subset is driven by user-specified timing constraints. This scheme differs from any previous approach, in which the entire design is always assumed to be power-gated.

In this new timing-driven paradigm, we use the timing constraints to determine what row can actually be gated, thus generating a mix of gated and non-gated rows. The choice of a row granularity clearly simplifies the physical design of such a mixed design, since it prevents the existence of both gated and non-gated cells in the same row. This power gating scheme share many similarities with well-known timing-driven power optimizations approaches such as multi-Vdd multi-Vt designs [13, 14], in which cells not on the critical path are replaced with slower, but more dynamic and/or static power-efficient ones, while meeting the original timing constraints. It is thus the combination of these two features (selective, timing-driven application of gating and row granularity) that make the proposed approach a true paradigm shift for power gating. Our methodology has the following advantages as compared to the existing approaches.

1. It allows to trade-off power versus performance by selecting (based on timing analysis) which rows can be power-gated. The main consequence of this trade-off is the possibility of achieving leakage savings at no performance penalty, which was not possible with previous approaches.

2. It allows, thanks to the use of a layout-friendly unit of gating, to fully automate the physical design of the power gating, and in particular the placement of the sleep transistors and the routing of the virtual ground lines. All existing approaches use a non-layout-aware granularity a generic cluster and hence makes the post-layout insertion and design closure extremely difficult.

3. It implicitly provides an exploration framework in the power/area/delay design space rather than just a single power-optimal solution under a given delay constraint. In fact, the gating of a row results in a increase in delay of all the cells in that row; the amount of delay, however, depends on how large is the sleep transistor used for gating it. Therefore, the increase in delay is affected by the area constraint. Notice the difference with respect to, e.g., multi-Vt design, in which the increase of delay of the replaced gate is fixed (i.e., the delay of the same cell with low-Vt). This trade-off was never addressed by existing methods, and as a consequence required the gating of the entire design no matter how large the sleep transistor size is to abide by the input timing constraints.

While the first two advantages have been also emphasized in the work of [12], the latter one is explicitly analyzed and exploited in this work for the first time. From the methodology point of view, this area/delay/power trade-off can be described by two quantities associated to a timing semantics:

Global Delay Degradation ($\alpha$) : This is a design specification, and defines the maximum allowable delay increase in the design/logic block. This is in fact the increase in the original critical path delay permitted when the design/logic block is power-gated. We denote this parameter by $\alpha$, expressed as percentage increase in the original critical path delay.

Gate Delay Degradation ($\beta$) : This is the delay increase of individual gates due to gating of a logic block. This increase is determined by the virtual ground voltage ($V_{GND}$) experienced by the gates in the design/logic block to be power-gated, and it depends on (i) the sleep transistor size and (ii) the peak current.
discharge of the design/logic block. For a given peak current value, the lower (higher) the sleep transistor size, the higher (lower) its on-resistance, and hence the higher (lower) the virtual ground voltage. This implies that individual gates will then experience higher (lower) delay degradation.

We denote this value by $\beta$, expressed as a percentage increase of the original gate delay.

In previous approaches, since gating the entire design implies gating every cell, $\alpha$ and $\beta$ must coincide, and therefore no control on the size of the sleep transistor is possible. As an example, if $\alpha$ is, say, $5\%$, all gates will also exhibit a $5\%$ delay penalty (and the sizing of the sleep transistor will be automatically determined).

In our methodology, these two quantities are decorrelated: for a given sleep transistor size (area constraint) and a given timing constraint we may have different values of $\alpha$ and $\beta$. There is however a maximum value for $\alpha$ and $\beta$ corresponding to a maximum allowable value for $V_{\text{VGND}}$ set by signal integrity considerations.

Now we will demonstrate the trade-off analysis made possible by the proposed approach on an example (the c5315 circuit from the ISCAS suite). In figure 1, the left first plot shows the area/delay trade-off (sleep transistor size vs. global timing constraint $\alpha$). Assume that we allow $\alpha = 5\%$ and area constraint 0.22 mm (expressed in terms of the maximum size of the sleep transistor – $W_{\text{sleep}}$). The shaded region shows the feasible region with respect to area constraints. As we see from the figure, for alpha = 5%, when any of the previous techniques are applied, the size required with this timing constraint falls outside the feasible region.

The right plot shows the same trade-off curve, this time applying our methodology and playing with different values of $\beta$. This is visible as a set of discrete point along a vertical line at $\alpha = 5\%$. We notice that there are several solution points some of which fall in the feasible region, and satisfy both timing and area constraints.

![Figure 1: Timing and Area constraints in Power Gating.](image)

As an additional result of the exploration, we can obtain the percentage of the design which can be power-gated with different sleep transistor sizes for a given value of $\alpha$. For this example, Figure 2 shows that for an area constraint of 0.22 mm sleep transistor size, and $\alpha = 5\%$ we can power-gate 78% of the design.

4. Clustering Methodology

4.1 High-Level Flow

In this Section, we first propose the high-level clustering flow as shown in the Figure 3 with input timing and area constraints. Since the clustering algorithm is based on values of $\alpha$ and $\beta$, we must translate the specified timing and area constraints into values of $\alpha$ and $\beta$.

The timing constraint is given as a maximum delay, and therefore directly translates into $\alpha$. The area constraint, conversely, is specified as a maximum value of the sleep transistor size denoted by $W_{\text{sleep-max}}$; in order to translate this quantity into a proper value for $\beta$, we first obtain the equivalent sleep transistor on-resistance $R_{\text{on-max}}$ using the following equation 1 from [7].

$$R_{\text{on}} = \frac{L}{\mu_n \cdot C_{\text{ox}} \cdot W_{\text{sleep}} \cdot (V_{\text{dd}} - V_{\text{th}})} = \frac{V_{\text{VGND}}}{I_{\text{peak}}}$$

Choosing a proper value for $\beta$ corresponding to the given area constraint and hence $R_{\text{on-max}}$ is difficult because of the following reasons. For each value of $V_{\text{VGND}}$, we maintain a look-up table which provides a corresponding value for $\beta$ and vice-versa. The virtual ground voltage $V_{\text{VGND}}$ depends on the peak discharge current of the cluster as given by $V_{\text{VGND}} = R_{\text{on}} \cdot I_{\text{peak}}$.

The value of $I_{\text{peak}}$, however, cannot be computed, since we do not know the cluster a priori. The cluster, in fact, consists of at least one layout row, but can include more than one row (or, in other terms, a sleep transistor can be shared among multiple rows). Without a realistic estimation of $I_{\text{peak}}$, we cannot compute the $V_{\text{VGND}}$ value; as a consequence, we don’t have a value for $\beta$.

In order to break this cyclic dependence between $I_{\text{peak}}$ and $\beta$ parameters, we assume that the cluster initially consists of the entire design (i.e., all the rows of the design). In this way, $\beta$ coincides

1In practice a look-up table is maintained with $R_{\text{on}}$ values for corresponding input $W_{\text{sleep}}$ due to a much more complex relation between the on-resistance and $W_{\text{sleep}}$. [Image 360x333 to 485x333]
with α and a corresponding value for $V_{\text{GND}}$ is read back from the look-up table. Also, since the cluster consists of the entire design, $I_{\text{peak}} = I_{\text{design}}$ and the initial $R_{\text{in}}$ value is computed by $R_{\text{in},\text{ini}} = V_{\text{GND}}/I_{\text{design}}$.

If we have enough area to power-gate the entire design i.e. $R_{\text{on},\text{max}} < R_{\text{on},\text{ini}} (W_{\text{sleep-max}} > W_{\text{sleep-ini}})$, then we power-gate the entire design. Conversely, if the area constraint does not allow it, then we have $R_{\text{on},\text{max}} > R_{\text{on},\text{ini}}$. We thus need to find an optimal cluster such that $R_{\text{on},\text{max}} \leq R_{\text{on},\text{ini}}$. To do this, we start increasing $V_{\text{GND}}$ and hence β values so that $R_{\text{on},\text{ini}}$ starts to increase. Since we increase the β value, some of the rows with timing critical gates might also get removed from the cluster, with the result of a decrease of $I_{\text{peak}}$, which further increases $R_{\text{on},\text{ini}}$.

Each time we set a new value to $V_{\text{GND}}$ and hence β, we run the clustering algorithm as shown in the figure 3 to find an optimal cluster of gates. We stop iterating when we reach a value of $V_{\text{GND}}$, and hence β, such that $R_{\text{on},\text{max}} \leq R_{\text{on},\text{ini}}$ which means $W_{\text{sleep-ini}} \leq W_{\text{sleep-max}}$. This check is done by the Area-check block in figure 3.

For properly sizing the sleep transistor at each iteration, we need to find the peak current discharged by the cluster $I_{\text{peak}}$. We use the methodology proposed in [11] for peak current estimation. The authors in this article propose an efficient methodology where they build temporal current waveforms for each gate based on the possible time intervals during which the gate can switch. Using this waveform for each gate, peak current for the entire design/cluster can be built and the current peak can be obtained. The block Peak current estimation and sleep transistor sizing shown in the flow does this function.

This iterative loop of our flow indeed will converge and this can be explained as follows. In the worst case scenario, when the area constraint is set to 0, as we go through the iterations, we increase the β value gradually during each iteration and finally after several iterations, the β will be so high that the cluster will be an empty set and the loop terminates.

### 4.2 Clustering algorithm

In this Section we describe our clustering algorithm with two input parameters α and β. As explained before, the algorithm uses layout rows as atomic candidate blocks for power gating. The main idea behind the clustering algorithm is to select for power gating a sub-set of rows (gates) with input parameters α and β values which set the constraints and maximizes a leakage cost function. The clustering algorithm used to solve this constrained optimization problem consists of two phases: A pre-processing step, and the actual clustering. In the following, we will describe in detail the two phases.

#### 4.2.1 Pre-processing Phase

This phase starts with a placed design, which can be abstracted as a set of rows $R = (r_1, ..., r_N)$. Let $(L_1, ..., L_N)$ denote a set of real numbers, corresponding to the leakage power of the rows. The rows of the layout can have very different leakage values depending on the type and number of cells in the row. We then need to extract the timing information (path delays) of the design to set the timing constraints. Using a timing analysis engine in the inner loop of the clustering algorithm would be too expensive in terms of computation time. Thus, we perform an initial accurate analysis with a commercial timing analysis engine where path delays and gate slacks can be extracted. Based on this initial information, we update the timing information during clustering, performing all necessary book-keeping operations on internal data structures.

As indicated by [7], there are two issues with path-based optimization: (i) The number of paths in a design is exponential in size. (ii) When a sub-set of rows are power-gated, critical path of a non-power gated design may not be the critical path of the power-gated design. To overcome these two problems, we use the heuristic commonly used in Static Timing Analysis as described in [7] and [12] to extract initial timing information of the design. We use a standard timing engine (PrimeTime© by Synopsys) to first extract the longest timing path through each cell in the design. Let us denote this set as the critical path set $\Pi$. For each path $p_i, i = 1, ..., m$ in $\Pi$, we maintain information on the path delay of the path and the cell instances in that path. Let the path delays of the paths in $\Pi$ be denoted as $p_{di}$. We see that a feasible cluster is one where if the rows in the cluster are power-gated and do not increase the path delay of any of the path in the path set $\Pi$ by more than $D_{\text{new}}$, where $D_{\text{new}}$ is the critical path delay of the power-gated design. If the critical path delay of the non-power-gated design is denoted by $D_{\text{old}},$ then $D_{\text{new}} = D_{\text{old}} + (\alpha * D_{\text{old}})$.

#### 4.2.2 Clustering Algorithm

We present two clustering algorithms, with different optimality vs. efficiency trade-offs. The first algorithm is an optimal one, and derives from casting of our constrained optimization problem into a 0-1 or Binary Integer Programming (BIP) model. The scalability of the BIP algorithm can become critical for extremely large designs since BIP is conventionally classified as NP-hard, thus making it difficult for usage in a design space exploration loop. Hence, we propose a linear time greedy heuristic to find a near optimal solution based on the timing criticality of the rows.

#### 4.2.3 0-1 or Binary Integer Programming

In this formulation, the solution is modelled as a binary vector where each bit represents a row; if a bit is set to one, the corresponding row is chosen for clustering, while if the bit is set to zero, the row is not clustered. Assuming the cardinality of the rows is N and the cardinality of the critical path set $\Pi$ is M, the BIP model for the constrained optimization problem is shown in figure 4.

![Figure 4: Binary Integer Programming (BIP) formulation of the problem](image-url)

The objective function is to find a feasible row sub-set which obey the timing constraints while maximizing the aggregate leakage power. Since we maximize the aggregate leakage power, high leakage rows will be power-gated which reduces the overall leakage power of the design. Here $x_i$ for $i \in \{1, ..., N\}$ are binary variables, denoting the rows to be either selected (value of 1) or not (value of 0). The values $a_{i,j}$ indicates the increase in path delay of a path $j \in \Pi$ by power gating the row $i$. This is calculated by first finding the gates in the row $i$ and which are on the path $j$, and then summing up the increase in delay of those gates. It can be expressed as follows. Let $d_k$ for $k \in \{1, ..., g\}$ denote gate delays of $g$ gates on row $i$ and on path $j \in \Pi$ and if $d_{k,i}$ for $k \in \{1, ..., g\}$ denote the corresponding increase in gate delays due to power-gating, such that $d_{k,i} = d_k * \beta$, then $a_{i,j} = \sum_{k=1}^{g} d_{k,i}$. Finally, the bound $b_j$ for $j \in \{1, ..., M\}$ indicates the available slack for each path $j$ in the critical path set $\Pi$. The value for this is computed as follows. $b_j = D_{\text{new}} - p_{di}$.
for \( j = \{1, \ldots, M\} \) where \( pd_j \) is the path delay of the path \( j \) and \( D_{new} \) is the new critical path delay as defined before. Note that there are as many constraints \((M)\) as the number of paths in the critical path set \( \Pi \).

4.2.4 Heuristic Algorithm

Figure 5 shows the pseudo-code of the clustering algorithm. It takes the information computed in the pre-processing phase and returns a Boolean vector \( Solution \) with one element for each row. \( Solution[i] = \text{TRUE} \) means that row \( i \) is included in the solution. Intuitively, the space to be explored contains \( \sum_{i=0}^{N} \binom{N}{i} = 2^N \). Each term of the sum represents the number of combinations of \( i \) elements (rows) out of \( N \). Since the space to be explored is exponential in size, complete enumeration is impossible for large values of \( N \) and hence necessitates for a lower run-time heuristic.

The actual core of the algorithm is the procedure \( \text{CHECK FEASIBILITY} \), which checks if a sub-set of rows abide by the timing constraints. The procedure keeps an array \( \sigma \) of book-keeping variables, one for each of the paths in \( \Pi \) determined in the pre-processing phase. Given a set of rows belonging to a cluster, for each path in \( \Pi \), (Line 1) and for each row in the cluster (Line 2), we compute the total aggregate delay degradation (Line 4) on the corresponding path by power-gating the sub-set of rows in the cluster. Note that as the total aggregate delay degradation (Line 4) on the corresponding \( \Pi \)-phase. Given a set of rows belonging to a cluster, for each path in \( \Pi \), (Line 1) and for each row in the cluster (Line 2), we compute \( \sigma[i] = \text{ TRUE, } x_i = 1 \)

\[
\sigma[i] = \alpha_{i,j} \times x_i
\]

(Line 13), we consider this solution as feasible and return TRUE. In the main algorithm \( \text{HeurCluster} \), we start with an initial solution consisting of all rows selected (Line 1). We then perform row-ranking where we sort the set \( R \) rows in decreasing order of timing criticality (Line 2). The timing criticality \( Tc \) of each row is computed as follows. Let \( \text{slack}_k \) for \( k = \{1, \ldots, g\} \) denote gate slacks of \( g \) gates on row \( i \). Then \( Tc_i = \sum_{k=1}^{g} \frac{\text{slack}_k}{g} \). Since gates on slower paths (critical paths) have smaller slack values, \( \frac{1}{\text{slack}_k} \) am-

![Figure 5: Heuristic Clustering Algorithm.](image)


5. Experimental Results

We have applied our clustering algorithms to a set of benchmark circuits taken from the ISCAS89 and MCNC suites, as well as to some other commercial design suites. Each benchmark was synthesized and placed using a 65nm CMOS technology library from STMicroelectronics and using Synopsys Physical Compiler for optimal timing. To facilitate the comparison of our methodology with the previous techniques, the timing constraints are set as in [6] which is \( 5\% \) \( \alpha \). To set the area constraints, we first compute the total size of the sleep transistor by using the methodology of [6], we keep this size as the upper bound denoted by 1X. We then apply our methodology in case we have the same \( \alpha \) value but only half (1/2) and one-fourth (1/4) of the upper bound sleep transistor size. Table 1 shows the results with the above settings. \( Benchmark \) shows benchmark circuit \( Gates \) and \( Rows \) show the number of gates and rows in the placed design. \( VGND \) and \( \beta \) show virtual ground voltage (mV) and gate delay degradation in \% achieved after a number of iterations \( Ite \) of our flow presented in figure 3. \( Gated \) and \( Savings \) show the percentage of gates power-gated and the leakage power savings computed using the methodology proposed in [12]. The results are reported for the BIP (BIP) solved using [15] and heuristic algorithms (Heur).

As we see from the results, for all the designs, having 1/2 the area used to power-gate the design, we can still power-gate more than 70\% of the design with maximum savings being achieved for switch_15x15 with 93.4\%. With 1/4 the area, we can achieve more than 60\% of the design being power-gated for all the benchmarks, with c5315 having the maximum leakage savings. This clearly shows that even clustering at row level granularity, we achieve very encouraging results. This can be attributed to most of the rows having gates with good amount of slack, which allows us to have higher \( \beta \) compared to \( \alpha \) thus achieving excellent results even with tight area constraints. We also report results where the timing constraint \( \alpha \) is set to zero meaning that no speed degradation of the power-gated design. For this setting, \( \beta \) is set at 5\% and the results are shown without any area constraints to evaluate how much of the design can be power-gated with zero value for \( \alpha \). We see that for some designs such as s38417, we can not power-gate the design with zero \( \alpha \) due to gates on the critical paths present in every row of the design.
Column **Speed-up** shows the ratio of runtimes of the BIP and the heuristic algorithms. We ran both the algorithms on Intel Pentium 4, 2.4GHz machine. As we can see, the heuristic algorithm achieves near optimal solution as compared to the BIP with excellent runtime-speed up for most of the designs. For very large benchmarks, the last two rows in the table, the BIP did not converge in reasonable amount of time and hence we don’t report any results for this. This fact further proves that the heuristic algorithm is computationally very efficient compared to BIP.

### 6. Conclusions

In this paper we have proposed a novel and a very flexible methodology for power-gating a design driven by timing and area constraints in contrast to only timing constraint driven techniques proposed in the earlier works. In particular we have proposed two algorithms for row-based power-gating methodology with different efficiency-accuracy trade-off and we have applied our methodology to a number of commercial and academic benchmark circuits. We see that the results on almost all the benchmark circuits are excellent further proving the effectiveness of our methodology. Based on encouraging results we have achieved by our clustering methodology, we can further improve the leakage power savings by using dual Vt technique concurrently with sleep transistor insertion.

### 7. REFERENCES


