Physically Clustered Forward Body Biasing for Variability Compensation in Nanometer CMOS design

Ashoka Sathanur
Politecnico di Torino ,
Ecole Polytechnique Fédérale
de Lausanne

Antonio Pullini
Ecole Polytechnique Fédérale
de Lausanne

Luca Benini
Università di Bologna

Giovanni De Micheli
Ecole Polytechnique Fédérale
de Lausanne

Enrico Macii
Politecnico di Torino

ABSTRACT

Nanometer CMOS scaling has resulted in greatly increased circuit variability, with extremely adverse consequences on design predictability and yield. A number of recent works have focused on adaptive post-fabrication tuning approaches to mitigate this problem. Adaptive Body Bias (ABB) is one of the most successful tuning "knobs" in use today in high-performance custom design. Through forward body bias (FBB), the threshold voltage of the CMOS devices can be reduced after fabrication to bring the slow dies back to within the range of acceptable specs. FBB is usually applied with a very coarse core-level granularity at the price of a significantly increased leakage power. In this paper, we propose a novel, physically clustered FBB scheme on row-based standard-cell layout style that enables selective forward body biasing of only of the rows that contain most timing critical gates, thereby reducing leakage power overhead. We propose exact and heuristic algorithms to partition the design and allocate optimal body bias voltages to achieve minimum leakage power overhead. This style is fully compatible with state-of-the-art commercial physical design flows and imposes minimal area blowup. Benchmark results show large leakage power savings with a maximum savings of 30% in case of 5% compensation and 47.6% in case of 10% compensation with respect to block-level FBB and minimal implementation area overhead .

1. Introduction

As a consequence of the scaling of device dimensions to nanometer size, we face a multitude of challenges in designing complex giga-scale systems. Major challenges come from parametric variation (intra and inter die variation of channel length, oxide thickness, doping concentration etc) [1], increased power-density leading to higher chip temperatures and circuit aging [3]. These phenomena have a direct and profound impact on system performance resulting in parametric yield loss and reduced system lifetime.

Focusing on variability, numerous approaches have been proposed to combat these effects and to increase parametric yield as well as tolerance to aging-induced variability. Statistical optimization approaches [2] target process variations as they select optimum design-time parameters (such as V_{th}) to maximize timing yield. Albeit effective, these approaches tend to conservatively constrain the design and rely on the accuracy of the statistical characterization of fabrication processes, which may not be known with high precision. As argued in [6], post silicon tuning can complement and sometimes outperform pre-silicon statistical optimization as it provides opportunity to tune individual dies to meet timing and other performance constraints. Post-silicon tuning can also address time-dependent variations, such as temperature-induced timing failures [4] NBTI-Induced transistor aging [3].

Among many post-silicon tuning strategies, ABB (Adaptive Body Bias) imposes limited overhead for body-bias generation and control [10], thus it is frequently deployed to combat the above mentioned effects which cause circuit timing failure [3, 8, 4]. ABB works on the principle of applying reverse bias voltage (RBB)/forward bias voltage (FBB) to increase/decrease the threshold voltage of devices. This results in increasing gate delay while reducing leakage in case of RBB and vice versa in case of FBB. One can trade-off delay with leakage to achieve low leakage or high speed.

In this work, we leverage FBB to speed-up designs which are slower than nominal due to process, temperature variations and circuit aging. However, we enhance current state-of-the-art solutions by allowing design-time (pre-silicon) specification of the "timing boost" made available for post-silicon tuning. The rationale is that FBB implies a large leakage penalty and that it must be used sparingly. Thus, if we expect low variations (e.g. if we are using a mature technology with reduced variability), or if our design has a small number of critical/slow paths, we do not need to forward body bias all the gates in a circuit block, but only a subset of them. Moreover, if one can afford to distribute more than one body-bias voltage, our technique can exploit additional degrees of freedom for maximizing delay sensitivity to FBB while minimizing the leakage power cost.

Given an amount of speedup required for a design, our methodology finds clusters in a fully placed netlist and allocates optimal FBB to them so as to improve the timing yield of the design. Our contributions can be summarized as follows:

- A post-placement, fine-grain (row-level) FBB implementation style, which gives precise control over timing and leakage power and very optimal tuning with low leakage overhead. Row level granularity greatly simplifies physical design since each row can be biased with a different body bias voltage with minimal area overhead as explained in section 3.3.
- Algorithms to find a minimum-leakage clustering for a given post-silicon delay boost target, under a user-specified constraint on the maximum number of body-bias voltages distributed on the design. An exact ILP-based formulation and a fast linear time heuristic are described and compared.
- Full integration of our approach with state-of-the-art commercial physical implementation flows and we demonstrate it on

978-3-9810801-5-5/DATE09 © 2009 EDAA

standard cell based designs using a 45nm industrial design kit. Our approach is fast and scales easily to large functional blocks with many thousands of gates. Experimental results show very significant leakage power reduction and precise control on the amount of speedup made available for post-silicon tuning with respect to standard un-clustered FBB.

2. Previous Work

A number of previous works have used ABB for compensating process variations to improve design yield. In [9], the authors analyze the effect of body bias on leakage components and provide a methodology to obtain optimal body bias voltage for leakage reduction and process compensation in nano-meter devices. However they don't provide results of applying ABB on complex designs. In [8] the authors use ABB (FBB and RBB) to reduce the impact of process variations in microprocessors. They maximize the die frequency under a given power constraint. They analyze different variants of applying ABB technique at different granularity (chip and block level). In [4], the authors propose to use ABB for temperature compensation and mitigate process variation. They propose to use ABB for compensating timing violations induced due to temperature variation and they propose novel algorithms to achieve this. In [5], the authors propose dynamic body biasing to meet timing. This methodology in contrast with static ABB uses in circuit timing sensing elements to sense timing of functional units dynamically at different operating workloads and automatically sets the body bias voltages there-by reducing leakage or increasing the operating frequency more efficiently compared to static ABB. However, all the above proposed techniques work at the block level granularity where each block receives a single body bias voltage. Our methodology in contrast to these techniques shows how one can apply ABB on even smaller granularity (clusters of gates in a block/design) there-by achieving higher leakage power savings but still meeting the required timing. Of the available literature on fine grained ABB, the work in [7] uses FBB to achieve active leakage power savings. They propose to use high V_{th} devices for synthesizing the circuit and then use FBB to speed up only a set of gates in the design to achieve original timing (synthesized with low V_{th}). The authors target to reduce the active leakage power by applying FBB but they do not compensate for timing variability. They also don't show any layout implementation of their methodology or discuss the underlying issues while we present a detailed analysis of layout issues in implementing FBB and also present a example layout with our FBB scheme.

The work in [6], propose to cluster the design, where each cluster is a sub-set of gates in a block, to which optimal body bias is applied to compensate for process variation. They propose a three-phase approach where in the first phase they obtain optimum body-bias voltage PDFs(probability density functions) for each gate and then they perform statistical aware clustering and finally they do post silicon tuning of the ABB clusters. This methodology has a few draw-backs. As indicated by authors in [6], for lower number of clusters, the clustering might resulting in dissimilar gates being clustered together resulting in higher leakage power cost. Since we perform clustering on finer granularity (row level), we can better tune the circuit compared to this method. For higher number of clusters, the area overhead due to placement perturbations and well separation due to adjacent gates having different body bias voltages becomes very large. In our methodology, since a unit of clustering is a single row, we don't encounter area overhead due to placement perturbations or adjacent gates having different body bias voltages.

3. Overview

In this section, we provide an overview of our design slowdown compensation methodology. First we describe the timing sensing methodology used to detect the slowdown in the design and then we discuss in detail the use of FBB to compensate the slowdown.

3.1 Post Silicon Tuning Calibration

In our methodology, to compensate the slowdown, we need to sense the timing of the circuit block to find if it is necessary to apply FBB and how much voltage is required to compensate the circuit slowdown. Process variation induced timing failures are static in nature and requires only one time compensation in contrast to temperature and circuit aging induced timing failures which are dynamic in nature. In order to track timing failures caused by dynamic effects, one has to periodically sense the circuit timing and then trigger a control circuitry which then generates optimal body bias voltage to be applied to the design under consideration. There are different ways in which timing sensing is done. In [5], the authors propose to use critical path replicas placed in different parts of the block and depending on their output, a control circuitry is triggered which then supplies the required body bias voltage to speed up the block. In [3] the authors discuss various techniques to sense the circuit timing. They propose to modify a standard flip-flop by inserting a timing monitoring circuit which detects significant shifts in the delay of the combinational logic whose output is connected to the data input of that flipflop. If any signal transition in the combinational logic block happens beyond a time T_{crit}, then the timing monitoring circuit flags a "timing alarm". In principle, if the path delays of the paths in a design are denoted by p_d, then if any of the paths have a degraded path delay greater than D_{crit}, where D_{crit} is the critical path delay, they can be considered as potential candidates which might violate the timing. Here the degraded path delay is given by $p_{d'} = p_d * (1 + \beta)$, where $\beta \leq 1$ is called the slowdown co-efficient which indicates the percentage by which each path in the design has slowed down. For example, If β = 5%, then all paths which have degraded path delays $p_{d'} = 1.05 * p_d$ greater than D_{crit} are paths which are potential candidates which might violate the timing. Note that the β value is set during design time depending on the amount of compensation we are targeting.

3.2 Forward Body Bias (FBB) Technique

Forward body bias (FBB) technique has emerged as an alternative solution to RBB since RBB worsens SCE (Short Channel Effects) and V_{th} variation across a die. It also increases the BTBT component of the leakage and hence its effectiveness diminishes as technology is scaled [14]. FBB is applied to a design with slow gates hence high V_{th} value to bring back the V_{th} to its target value. FBB methodology works on the principle of applying a positive voltage vbs_n to the body terminal of the NMOS transistor and a positive voltage of vbs_p to the PMOS transistor. For simplicity we denote the applied body bias voltage as vbs, which denotes a voltage of $vbs_n = vbs$ applied to NMOS and $vbs_p = Vdd - vbs$ applied to PMOS transistor. Applying FBB to both PMOS and NMOS devices requires a triple well fabrication process which is already in use in present day 45nm processes. To evaluate the effectiveness of FBB, we did spice simulations on a 45nm CMOS SOI process and the results for a simple inverter is shown in the figure 1. We report speed-up and leakage power increase achieved with respect to no body bias (NBB) for various forward body bias voltages applied. We see a linear increase in speed-up while an exponential growth in leakage power as expected. In our experiments, we applied a range of body bias voltages with vbs ranging from 0 to 0.95V (V_{dd}). We measured the delay change and the current consumed during off

state at the source terminal. Our experimental results show that beyond 0.5V of vbs, increased leakage and forward source-body junction current [14] limits the range of vbs to 0V-0.5V. So we restrict the body bias voltage to these values in all our experiments. From the figure 1, We also see that by applying the maximum body bias voltage we can achieve up-to 21% speed-up at the expense of 12.74X increase in leakage power.

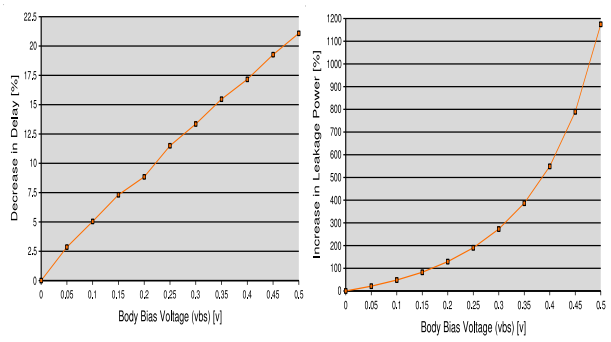

Figure 1: Delay and Leakage power variation of an inverter with varying body bias voltage in 45nm CMOS.

In general ABB (FBB and RBB) techniques have very low implementation area overhead which makes them preferred choice for post-silicon tuning. As reported in [8], applying ABB even at the block level granularity where each block can have access to multiple bias voltages (like in our methodology), the total area overhead for bias generation, required body bias buffers and routing of bias signals incurs 2%-3% of the total die area. One can achieve up-to a 32mV resolution from the body bias generator as described in [8]. In this paper we assume we have a 50mV resolution from the body bias generator. So we have 11 different vbs values at our disposal, for NMOS starting from 0 to 0.5V in steps of 50mV and for PMOS starting from 0.95 to 0.45 in steps of 50mV. Figure 2 shows a block diagram summarizing our tuning methodology. As shown in the figure, there are 4 circuit blocks in a design each having a T_{ci} (i indicating the block number) which indicates a timing violation in the block. The central body bias generator generates appropriate body bias voltages (in the figure, number of vbs=2) to compensate the slowdown in the block.

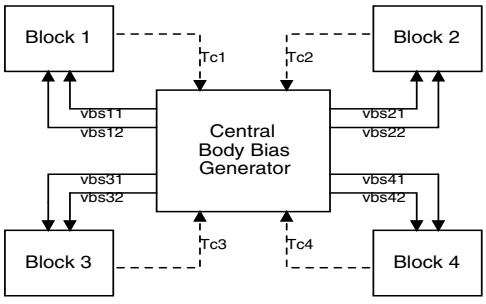

Figure 2: A block diagram showing FBB implementation methodology.

3.3 FBB implementation

In this section, we show how we have implemented FBB on a traditional standard cell based layout style. Figure 3 shows the layout before and after body bias. The layout with body bias shows two body bias voltages $vbs1$ and $vbs2$. The body bias signals are routed on top metal layer while the horizontal supply lines (Vdd,Gnd) are in metal 1 layer. Since we have two body bias voltages, we have two pairs of body bias signals. In each pair, one of the signal biases the NMOS transistors and the other one biases the PMOS. As we see from the figure, layout before body bias has its body bias contact cell connected to the supply lines which indicates no body bias being applied. In the layout with body bias, the body bias contact cells are placed and connected depending on the body bias voltage applied to the specific row. As shown in the figure, $Row1$ is connected to $vbs1$, so the contact cell in $Row1$ is placed right below the $vbs1$ lines. Similarly $Row2$ is connected to $vbs2$, so the contact cell in the $Row2$ is placed right below the $vbs2$ lines. The design rules require the body bias contact cells to be placed every 50um (in the technology we have used) for proper biasing of the body. We incur a maximum 6% increase in utilization on each row when we have two body bias contact cells every 50um. Since there is good amount of spatial slack available on each row of the design, placing the contact cells will only increase the row utilization but does not incur any additional area overhead. However having more than two body bias contact cells might result in too high row utilization forcing an increase in die area. As a consequence, we restrict the number of vbs to no more than two. Thus, we can have a maximum of three clusters corresponding to no body bias, body bias 1 and 2. We also don't need any well separation on each row, since all the adjacent gates on a row receives the same body bias voltage. However we incur a very small overhead due to well separation required when adjacent rows in the design are assigned to different body bias voltage as shown in figure 3.

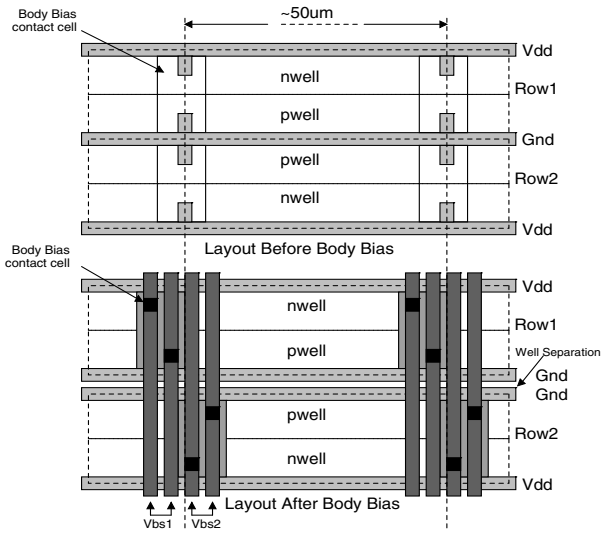

Figure 3: An abstract view of a portion of the standard cell layout with FBB.

4. Algorithms for Optimal FBB allocation

In this section, we present two algorithms for optimal body bias allocation, the first one is an exact solution where we cast the problem into an ILP to find the optimal clusters with their respective body bias voltages. In the second algorithm, we propose a two pass linear time heuristic to solve the same problem. In both the cases, the body bias clustering problem can be defined as follows. *Given a placed design with a set of rows, partition the design into C clusters (each with a sub-set of rows), each with its own body*

bias voltage such that the overall timing is met while minimizing the leakage power. Note that our methodology works on standard cell based designs and we apply body bias voltage at standard cell row level granularity. So each row in the placed design forms the unit element or lowest granularity at which body bias is applied .

4.1 Pre-Processing

We start with a placed design, which can be abstracted as a set of N rows $R = (r_1, ..., r_N)$. Let us assume that we have P body bias voltages available from a body bias generator. These P body bias voltages form P possible clusters to which each row of the design can be assigned. We define C as the number of clusters we need to partition the design. So $C \leq P$. Note that this constraint is required since it might not be possible to provide all possible P body bias voltages to each row in the design due to implementation limitations as explained in section 3.3. As described before, in this work, we assume that we have a body bias generator which generates voltages at 50mV resolution and since we have a maximum of 0.5V as vbs, we have P = 11 body bias voltages. For each of these voltages, we compute the average leakage power of all the rows in the design. Let $L_{i,j}$ denote the average leakage power of the ith row assigned to jth body bias voltage for $i \in 1, \ldots, N$ and $j \in 1, \ldots, P$.

In the pre-processing phase, we compute these values, and then we extract the timing information (path delays) of the design which form the timing constraints. To overcome the problems of path based optimization as discussed in [11], we use the heuristic as described in [11] to extract the timing information of the design. We use a standard static timing analysis engine (PrimeTime© by Synopsys) to extract *the longest timing path through each cell* in the design. We prune this set to end up with a unique set of paths $\Pi = (p_1, \ldots, p_M)$. For each path $p_i \in \Pi$, we maintain information on the path delay of that path (p_i) and the list of cell instances along that path. D_{crit}.

4.2 Optimal FBB allocation algorithm

In this section, we cast our problem into an ILP where the objective function is to minimize the total leakage power one has to spend in order to speed up the design and hence obey the timing constraints. The design is partitioned into C clusters to achieve this goal. The set partitioning problem can now be stated as the following ILP:

$$\textbf{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{P} x_{i,j} * L_{i,j} \tag{1}$$

Subject to

$$\sum_{i=1}^{N} \sum_{j=1}^{P} a_{i,j,k} \cdot x_{i,j} \leq b_k, \quad \text{for } k \in \{1, \ldots, M\} \tag{2}$$

$$\sum_{j=1}^{P} x_{i,j} = 1, \quad \text{for } i \in \{1, \ldots, N\} \tag{3}$$

$$\left.\begin{array}{l} \sum_{i=1}^{N} x_{i,j} \leq F * y_j, \quad \text{for } j \in \{1, \ldots, P\} \\ \sum_{j=1}^{P} y_j \leq C \end{array}\right\} \tag{4}$$

$$x_{i,j}, y_j = \{0 \; or \; 1\}, \quad \text{for } i \in \{1, \ldots, N\}, j \in \{1, \ldots, P\} \tag{5}$$

The problem is expressed in terms of the binary variables $x_{i,j}$, where $x_{i,j} = 1$ indicates that row i is assigned to body bias voltage j. Equation 1 is the objective function, which expresses the minimization of the total leakage power in the design. The constraint of 2 expresses the timing constraints relative to the critical path set Π. The value $a_{i,j,k}$ denotes the reduction in path delay of a path $k \in \Pi$ when FBB is applied to the row i with a body bias voltage of vbs_j. This is calculated by first determining the gates in the row i that are on the path k and with a body bias voltage of vbs_j and then summing up the reduction in delay of those gates. This can be done as follows. Let $Q_{i,k}$ be the number of cells on row i and on path $k \in \Pi$, and $d_l, l \in \{1, \ldots, Q_{i,k}\}$ the delays of these gates. When applying body bias voltage to row i with a vbs of vbs_j, the delay of these gates will reduce by a quantity $\delta_l, l \in \{1, \ldots, Q_{i,k}\}$. Then, $a_{i,j,k} = \sum_{l=1}^{Q_{i,k}} \delta_l$. Note that there are P different body bias voltages leading to P such coefficients for each row and for each path.

The right hand side of the constraint b_k for $k \in \{1, \ldots, M\}$ indicates the speed-up required for each path k in the critical path set Π. This value is computed as $b_k = D_{crit} - (p_k * (1 + \beta))$ for $k = \{1, \ldots, M\}$. Note that there are as many constraints (M) as the number of paths in the critical path set Π. Constraints 3 says that a row can belong to one cluster only, out of the possible P different clusters.

Inequalities 4 define the constraints for which only C or less clusters or body bias voltages are allowed out of P possible values. This is done by introducing auxiliary variables [12] in the formulation. We introduce P such variables y_js, each for one cluster. Note here that F is a very large number. Finally, Equation 5 define the bounds for $x_{i,j}$ and y_j variables, which are binary variables.

4.3 Heuristic approach to FBB allocation

In this section, we propose a linear run-time heuristic, to solve the FBB clustering and allocation problem. Its a two-pass greedy algorithm based on timing sensitivity of the rows in the design. In the first pass of the algorithm, we try to find the timing feasible solution and in the second pass, we try to minimize the leakage power within this timing feasible space. The two pass algorithm is as shown in the figure 5. The core of the algorithm is the *CheckTiming* routine as shown in figure 4. It performs a timing check given a *Solution*. A *Solution* consists of binary variables each for one row of the design which specifies which row of the design is assigned to which body bias voltage. For each path in Π *Line1*, We compute and sum the co-efficients $a_{i,j,k}$ for all the rows in the design *Lines 2-7* as described in section4.2. We then check if we are within the timing bounds for each of these paths *Lines 8-11* and report FALSE if we violate the bound for any of the paths or TRUE if none of the paths are violated.

The heuristic algorithm begins with the first pass *PassOne*, where we try to find a timing feasible solution by assigning all the rows to a particular body bias voltage. We begin with no body bias voltage assigned to all the rows. Then for each higher body bias voltage, which reduces the delay of all the gates by a certain amount *Line1*, We assign all the rows to that body bias voltage *Lines 2-4* and perform the timing check (*CheckTiming*) *Line5*. If the timing is violated we move to the next higher body bias voltage and hence speed-up the gates by a larger amount and perform the timing check, and we do this until we find a timing feasible solution. We report FALSE if none of the body bias voltages assigned to all the rows lead to a timing feasible solution. Once we find the body bias voltage which gives us the timing feasible solution, we return this voltage value denoted by j_{opt}. Note that this is a timing feasible solution but has very high leakage power due to low threshold voltage assigned to all the gates in the design.

In the second pass of the algorithm, *PassTwo* ,from the solution ob-

tained from the *PassOne*, we try to find a timing feasible solution but with a lower leakage power value. The intuition behind this is, there are rows in the design which have gates in the non critical paths and hence can tolerate a higher threshold voltage or lower body bias voltage and hence will result in reduction of leakage power. The *PassTwo* begins with ranking the rows in the increasing order of timing criticality (*Perform Row Ranking*) *Line 2*. We compute the timing criticality co-efficient to rank the rows as follows. For any given row in the design i, Let $Q_{i,k}$ the number of cells on row i and on path $k \in \Pi$. Then the timing criticality co-efficient c_{ti} of the row i is calculated as $c_{ti} = \sum_{k=1}^{M} \sum_{i=1}^{Q_{i,k}} 1/slack_k$. $slack_k$ denotes the slack on path k. After performing row ranking, we then start from the j_{opt} body bias voltage, and we start dropping the least timing critical row to the next lower body bias voltage *Lines 9-10*. Once we have a timing violation reported by *CheckTiming* by moving a row i to a lower body bias voltage, we move back the row i to the original body bias voltage *Lines 11-13*. We don't drop any further rows since a row lower in ranking than i is more timing critical and hence will lead to a timing violation. So all the rows with a ranking lower than i including the row i now form a cluster with a body bias voltage j_{opt}. We lock all the rows in the cluster. We then increment the *clustercount* variable which keeps track of number of clusters formed *Line 14*. We start the next iteration with the remaining set of un-locked rows, we repeat the same steps from *Line 9-14*. We also make sure that the number of clusters formed is less than or equal to C *Line 5*. The cost of the heuristic algorithm can be computed as follows. The cost of every iteration of the algorithm is the cost of running the *CheckTiming* routine. The total cost of the heuristic is $Tot_{cost} = C_{pass-one} + C_{pass-two}$. $C_{pass-one}$ is a constant number and the cost of $C_{pass-two} = O(P * N)$. So the total cost $Tot_{cost} = O(P * N)$ or $O(N)$. So the run-time of the heuristic algorithm is linear in the number of rows in the design.

```
CHECKTIMING(Solution,R,Π) {
1    foreach path k = 1,...,M ∈ Π
2        foreach row i = 1,...,N
3            if (Solution[i,j] = TRUE, x_{i,j} = 1)
             // for each row included in solution
4                σ[k]+ = a_{i,j,k} * x_{i,j};
5            endif
6        endfor
7    endfor
8    foreach path k = 1,...,M ∈ Π
9        if (σ[k] > b_j) return FALSE;
10       endif
11   endfor
     return TRUE;
}
```

Figure 4: Check Timing Routine.

5. Experimental Setup and Results

We applied our FBB allocation methodology to a total of nine designs; five of them are public-domain benchmarks taken from the ISCAS benchmark suite, while the remaining are circuit modules of an industrial SoC. Each design was synthesized and placed using a reduced library of gates consisting of inverters, and, or, nor, nand and D-flip-flops of different drive strength. We used the 45nm CMOS technology library from STMicroelectronics and mapped the design using Synopsys Physical Compiler for optimal timing. For each of the gates in the library, we characterized its delay increase and average leakage power for different body bias voltages.

```
PASSONE( R, Π, L) {
1    foreach body bias j = 1,...,P
2        foreach row i = 1,...,N
3            Solution[i,j] = TRUE, x_{i,j} = 1;
             // Assign all rows to body bias voltage j
4        endfor
5        while (CHECKTIMING(Solution,R,Π))
6        j_opt = j;
7        return j_opt
8    endfor
9    return FALSE
}

PASSTWO( R, Π, L, j_opt, C ) {
1    BestSolution[i,j_opt]= (TRUE,...,TRUE);
2    Perform Row-Ranking;
     // sort R in increasing order of timing criticality;
3    Solution[i,j_opt]= BestSolution;
4    foreach body bias j = j_opt,...,2
5        if clustercount > C;
6            BestSolution=Solution;
7            break
8        endif
9        Solution[i++,j] = FALSE;
10       Solution[i++,j–] = TRUE;
         // put the row in lower body bias voltage
11       while (! CHECKTIMING(Solution,R,Π))
12           Solution[i++,j] = TRUE;
13           Solution[i++,j–] = FALSE;
         // swap back to original body bias voltage
14           clustercount+=1;
15       endwhile
16   endfor
17   BestSolution = Solution;
```

Figure 5: Two Pass Heuristic Clustering Algorithm.

Table 1, shows the experimental results. *Benchmark* column shows the circuit. *Gates* and *Rows* shows the number of gates and rows respectively in the design. β indicates the circuit slow-down co-efficient as described before in section 3.1. Column *Single BB* shows the leakage power spent when block level body biasing technique is applied as done by previous works. Here the entire design/circuit receives a single body bias voltage. We compare our methodology with this value. To compute this value, we ran only the *PassOne* of the Heuristic clustering algorithm. Columns *ILP* and *Heuristic* show leakage power savings in % achieved compared to *Single BB* by running the ILP and the heuristic algorithm for 2 and 3 clusters. Note that due to additional area overhead incurred in implementing more than 3 clusters as explained in section 3.3, we limit the maximum number of clusters to 3. Finally the column *No.Constr* shows the number of timing constraints in each of the problems solved thus indicating the complexity of the problems. Results show very large leakage power savings in both the heuristic and the exact solution cases compared to block-level FBB with a maximum of 30% in case of β=5% and 47.6% in case of β=10%. We see that the savings achieved is higher in case of higher β value for all the designs thus indicating the effectiveness of applying our methodology when the circuit slow-down is higher. We also see that the increase in savings achieved with $C = 3$ as compared to $C = 2$ is very marginal in most of the cases. To further investigate this, we ran experiments on c5315 benchmark from $C = 2$ to $C = 11$ for $\beta = 5\%$ and we saw a marginal increase in leakage power savings of 2.56%. This shows that one can implement a very low area overhead layout with few body bias voltages but still

Benchmark	*Gates*	*Rows*	β	*Single BB*	*ILP*		*Heuristic*		*No.Constr*
					$C=2$	$C=3$	$C=2$	$C=3$	
	[#]	*[#]*	*[%]*	*[uW]*	*[%]*	*[%]*	*[%]*	*[%]*	
c1355	439	13	5	0.17	11.76	17.65	11.76	11.76	32
			10	0.33	30.30	33.33	27.27	30.30	72
c3540	842	15	5	0.42	23.08	23.08	11.54	19.23	31
			10	0.82	40.82	44.9	30.61	34.69	70
c5315	1308	23	5	0.26	21.43	21.43	16.67	16.67	11
			10	0.49	46.34	47.56	31.71	36.59	33
c7552	1666	26	5	0.63	19.05	20.63	17.46	17.46	5
			10	1.23	44.72	47.15	30.89	36.59	11
adder_128bits	2026	28	5	1.43	26.57	30.07	23.08	25.17	26
			10	2.26	28.76	33.63	20.80	25.22	55
c6288	2740	33	5	1.74	4.60	5.17	3.45	3.45	773
			10	3.38	22.78	23.96	18.64	18.64	810
Industrial1	4219	41	5	3.07	20.85	24.76	16.94	18.57	136
			10	6.13	33.77	36.22	22.51	24.63	237
Industrial2	10464	63	5	5.83	-	-	8.58	8.58	489
			10	11.36	-	-	24.74	24.74	1502
Industrial3	23898	94	5	12.25	-	-	15.67	16.41	1012
			10	23.88	-	-	25.21	25.21	2867

Table 1: Experimental Results

achieve optimal savings. The increase in the area due to well separation required when adjacent rows are in different clusters was always below 5% for all the cases.

We also compare the savings achieved by the exact and the heuristic algorithms and we see that the heuristic solution is close to the optimal solution in most of the cases. We ran all our experiments on Intel 2.4GHz machine. We use the solver [13] to run all our ILP experiments. The run-times of the ILP algorithm was comparable to the heuristic in case of smaller designs while was significantly higher than that of the heuristic for larger benchmarks (speed-up of more than 1000X) thus proving the scalability of the heuristic algorithm. We report no ILP results for the Industrial2 and Inductrial3 benchmarks since the ILP did not converge in a specified amount of time. We implemented our row-based FBB on an example design C5315 and the complete placed and routed layout is as shown in the figure 6. Since the design was small, we could route only one set of body bias lines (4) corresponding to 2 vbs values through the center of the layout.

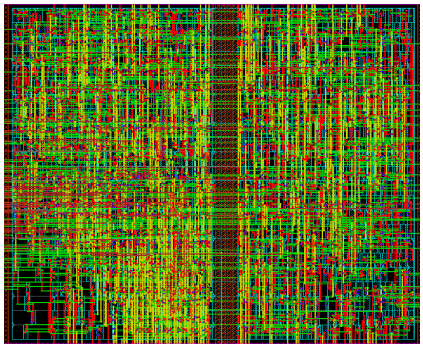

Figure 6: Placed and Routed C5315 design with 2 vbs.

6. Conclusions

In this paper, for the first time we have introduced the concept of physically clustered FBB methodology where we incur extremely low area overhead while achieving very high leakage power savings. This is possible due to having very fine granularity at which body bias can be applied (row-level). This not only helps us in very easy layout implementation but also provides very fine tunability of delay and leakage power. We proposed efficient algorithms for partitioning the layout into multiple clusters and allocating optimal body bias voltages to achieve the required speed-up while reducing the leakage power cost.

7. Acknowledgment

This work is supported by the European FP7 ICT-REALITY project.

8. REFERENCES

[1] S. Nassif "Delay variability: Sources, impacts and trends," *ISSCC,2000*, pp. 368-369.

[2] M. Mani, et al. ,"An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints", *Proc. of ACM/IEEE DAC-05*, pp. 309-314.

[3] S. Mitra. "Circuit Failure Prediction for Robust System Design in Scaled CMOS".*International Reliability Physics Symposium,* May.2008.

[4] S. V. Kumar, et al. , "Mathematically assisted adaptive body bias (ABB) for temperature compensation in gigascale LSI systems",pp. 559-564, *ASPDAC 2006*.

[5] R. Teodorescu, et al. , "Mitigating Process Variation with Dynamic Fine-Grain Body Biasing", *IEEE MICRO*, December 2007.

[6] S. Kulkarni, et al. , "Design-Time Optimization of Post-Silicon Tuned Circuits using Adaptive Body Bias", *IEEE TCAD*, Vol. 27, No. 3, March 2008, pgs. 481-494

[7] V. Khandelwal, A. Srivastava, "Active Mode Leakage Reduction Using Fine-Grained Forward Body Biasing Strategy", *Integration the VLSI Journal*, Vol. 40, No. 4, pp. 561-570, July 2007.

[8] J. Tschanz, et.al , "Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage", *IEEE JSSC*, pp. 1396-1402, November 2002.

[9] C. Neau , K. Roy, "Optimal Body Bias Selection for Leakage Improvement and Process Compensation over Different Technology Generations", *IEEE ISLPED-03*, pp. 116-121.

[10] S. Narendra, et al. , "Forward body bias for microprocessors in 130-nm technology generation and beyond" *IEEE JSSC*,Vol. 38, No. 5, May 2003.

[11] A. Ramalingam, and et al. , "Sleep transistor sizing using timing criticality and temporal currents", *ASPDAC-05*, pp. 1094-1097.

[12] Frederick S. Hillier, Gerald J. Liberman, " Introduction to Operations Research," *Eighth Edition*.

[13] ftp://ftp.es.ele.tue.nl/pub/lp_solve.

[14] "Leakage in Nanometer CMOS Technologies (series on Integrated Circuits and Systems)", Edited by S.G.Narendra, A.Chandrakasan, *springer-2006*.