7th RTD Framework Program

REALITY

Reliable and Variability tolerant System-on-a-chip Design in More-Moore Technologies

Contract No 216537

Deliverable D3.3

Report: Design Techniques for Enhancing Data Path and Controller Variability Tolerance

Version 2.0

Editor:	Andrea Acquaviva
Co-author / Acknowledgement:	Luca Benini V2.0
Status - Version:	V2.0
Date:	26/01/2009
Confidentiality Level:	Public
ID number:	IST-216537-WP3-D3.3.doc

The REALITY Consortium consists of:

Interuniversity Microelectronics Centre (IMEC vzw)	Prime Contractor	Belgium
STMicroelectronics S.R.L. (STM)	Contractor	Italy
Universita Di Bologna (UNIBO)	Contractor	Italy
Katholieke Universiteit Leuven (KUL)	Contractor	Belgium
ARM Limited (ARM)	Contractor	United Kingdom
University Of Glasgow (UoG)	Contractor	United Kingdom

1. Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

2. Acknowledgements

The editor Andrea Acquaviva and Luca Benini acknowledges contribution from Luca Benini, Miguel Miranda and Tom Tassignon.

3. Document revision history

Date	Version	Editor/Contributor	*Comments*
26/12/2008	V1.0	Andrea Acquaviva, Luca Benini	First draft
26/01/2009	V2.0	Andrea Acquaviva, Luca Benini, Miguel Miranda, Tom Tassignon	Final version

4. Preface

The scope and objectives of the REALITY project are :

- Development of design techniques, methodologies and methods for real-time guaranteed, energy-efficient, robust and adaptive SoCs, including both digital and analogue macro-blocks“

The Technical Challenges are :

- To deal with increased static variability and static fault rates of devices and interconnects.
- To overcome increased time-dependent dynamic variability and dynamic fault rates.
- To build reliable systems out of unreliable technology while maintaining design productivity.
- To deploy design techniques that allow technology scalable energy efficient SoC systems while guaranteeing real-time performance constraints.

Focus Areas of this project are :

- “Analysis techniques” for exploring the design space, and analysis of the system in terms of performance, power and reliability of manufactured instances across a wide spectrum of operating conditions.

- “Solution techniques” which are design time and/or runtime techniques to mitigate impact of reliability issues of integrated circuits, at component, circuit, architecture and system (application software) design.

The REALITY project has started its activities in January 2008 and is planned to be completed after 30 months. It is led by Mr. Bart Dierickx and Mr. Miguel Miranda of IMEC. The Project Coordinator is Mr Tom Tassignon. Five contractors (STM, ARM, KUL, UoG, UNIBO) participate in the project. The total budget is 2.899 k€.

5. Abstract

This report describes the design, exploration and experimentation of architectural techniques for data path and controller variability tolerance. The aim of these techniques is to automatically identifying static and dynamic variations, recalibrate the operation mode to adapt to these variations and finally to provide monitoring information to the system level. The exploration of state of the art strategies lead to the identification of two main solutions, namely Adaptive Voltage Scaling (AVS) and Adaptive Body Biasing (ABB). Both are aimed at reducing the performance impact of Vth variations by playing either with supply or body bias voltage. ABB is considered the most promising solution as it introduces limited area and leakage energy overhead compared to AVS for a given performance compensation level. In this report we first provide an overview of both techniques, then we focus on ABB by describing current implementation adopted and finally we introduce an innovative Forward Body Bias (FBB) solution. We also report on results on several benchmark circuits, including a few large datapah-with-controller benchmarks.

6. List of Abbreviations

REALITY	Reliable and Variability tolerant System-on-a-chip Design in More-Moore Technologies
CAD	computer aided design
DLC	
DMT	discrete multi-tone
DSP	digital signal processing
FFT	fast Fourier transform
HW	Hardware
IC	integrated circuit
MPSoC	Multiprocessor System-On-Chip
QoS	quality of service
SoC	system on chip
SOHO	small office/home environment
SW	Software

7. List of Tables

8. List of Figures

9. Table of contents

10. 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 such as intra and inter die variation of channel length, oxide thickness, doping concentration [1]. These phenomena have a direct and profound impact on system performance resulting in parametric yield loss and reduced system lifetime. 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 Vth) 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 [3], 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 [5].

Common techniques to compensate process variation induced performance deviations during post-silicon testing are: i) Dynamic adaptation of processor voltage; ii) Reverse/Forward body biasing. The first one exploits voltage supply increase to boost the circuit performance, while the second achieves the same effect by reducing the threshold voltage. Applying these techniques in an unconditioned fashion would introduce an unacceptable energy overhead. As a consequence, adaptive strategies have been developed, namely AVS [14] and ABB [15], aimed at applying voltage (either Vdd or Vth) adaptation only when on-line monitoring circuits signal that speed-up is required for maintaining correct functionality at the required performance levels. Moreover, adaptive techniques can react to time and temperature dependent performance variations.

While it has been shown in [12] that adaptive voltage scaling (AVS) requires a much smaller change (percentage-wise) in supply voltage than adaptive body biasing (ABB) requires in threshold voltage to achieve a target frequency boost of a processor core, and for this reason AVS in principle has a much milder impact on leakage and is a more power-efficient and thermally compatible solution than ABB, the area and power overhead introduced by the required level shifters between the multiple voltage domains make AVS less attractive than ABB in real implementations.

Indeed, ABB imposes limited overhead for body-bias generation and control [7], thus it is frequently deployed to combat the above mentioned effects which cause circuit timing failure [4, 5, 6] 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 project, UNIBO implemented both AVS and ABB techniques to investigate in depth the cost/benefits of both solutions. We developed two innovative AVS and ABB strategies: in this report we summarize the result of AVS exploration and while we focus on the description of an innovative ABB technique that will be proposed for integration in the target demonstrator developed in WP5, namely the “Physically Clustered Forward Body Biasing”. Variability characterisation information provided by WP2 for the reference target architecture of this project will be exploited to tune the proposed technique.

The rest of the report is organized as follows. In Section 11 the exploration of state of the art techniques for architectural level variability tolerance is reported. In Section 12 the FBB technique is described as well as the algorithm for optimal FBB allocation and the experimental validation. Section 13 explains future work in WP3 and concludes the report.

11. State of the Art in Architectural Level Variability Tolerance

As summarized by the survey in [15], three kinds of techniques have been proposed to enhance yield under variations while minimizing the design overhead. In the statistical design approach, circuit parameters are modeled as statistical distributions and the circuit is designed to meet a constraint on yield, typically acting on gate sizing or dual-Vth assignment [16, 17]. Alternatively, a variation avoidance approach aims at synthesizing circuits in such a way that delay failures due to variations can be identified at runtime and avoided by adaptively switching to two-cycle operations [18]. Finally, a number of techniques take the approach of post-silicon compensation and correction. In this case, parameter shifts are detected and compensated after manufacturing by changing supply voltage [13], frequency, body bias [14] or clock skew [19]. Under certain variation scenarios, adaptive pipelining techniques [19,20] have been proposed to avoid unacceptable operating frequency penalties.

11.1. AVS Techniques

In [21, 22] a global voltage controller adjusts the supply voltage by monitoring the error rate of the entire pipeline. Only in [23] each pipeline stage is provided with its own voltage controller and can thus be operated at its own optimized voltage. From an implementation viewpoint, the concept of operating coarse-grain design sub-blocks at a separate supply voltage is today common practice for low-power designs through a voltage island design style [24]. Early approaches applied this technique on a core-by-core basis [25]. Other researchers advocate for a finer granularity VI generation. On one hand, this reduces the overhead that ensues from raising the voltage of an entire core or unit only because of the presence of a small percentage timing critical cells in it. On the other hand, a logic-based generation of voltage islands constrains the physical placement [26]. For this reason, fine-grain voltage-islands derived through a placement driven synthesis framework were proposed in [27, 28].

A more radical approach has been explored in this project to placement driven VI generation, which aims at the minimum modification of performance pre-optimized placements through the grouping of physically adjacent cells into the same voltage island. Moreover, we precisely quantified the level shifter overhead and its impact on total system power and area for an industrial 65nm technology library. Nonetheless, we proved the viability of the voltage island approach to deal with process variations by assessing the power savings with respect to chip-wide supply voltage adaptation. The technique is presented in the following section.

11.2. Placement-Aware Multiple Voltage Island Design Style

This methodology assumes that a given RTL design has already undergone placement-aware logic synthesis driven by performance optimization directives. Once the global placement is determined and timing can be estimated with enough precision, the resulting netlist is fed to an additional design step where process variations are addressed. The objective at this stage is to minimally impact performance results achieved by the previous

physical synthesis run, while avoiding worst-case timing margins for process parameter variations.

The methodology involves two steps. At design time, variation-aware statistical static timing analysis (SSTA) allows to pre-characterize multiple scenarios of performance constraint violations in fabricated chips. For each scenario, placement-aware cell grouping algorithms are then applied to identify the minimum subset of adjacent cells to be operated at high-Vdd to tackle the timing violation through the correspondent performance boost.
Voltage islands are generated in such a way that moving from a violation scenario to a more severe one, only the supply voltage of 1 additional voltage island needs to be raised. After chip fabrication, timing sensing circuits integrated into the design to assess whether a given process variation scenario took place, and the most appropriate number of voltage islands is powered at high-Vdd as determined at design time.

We tested this placement aware methodology on a 4-way VLIW architecture and on a 65nm technology library. We evaluated the power savings it can achieve with respect to traditional full chip supply voltage adaptation techniques in spite of the overhead associated with level shifter insertion. Results show that this solution is more power-efficient than chip-wide Vdd adaptation from 8 to 27%.

The main drawback of this technique is the area overhead of level shifters. Even though all advanced technology libraries include level shifter cells to support signals across voltage islands, these cells have an area, timing and power cost. Level shifters are an affordable cost in coarse-grained core-by-core voltage island clustering. Our fine-grained approach does support re-synthesis with insertion of level shifters, but we found that preservation of the original cell placement becomes impossible. Iterative re-synthesis allowed us to converge to a clustered solution meeting the design constraints, but in our experiments significant designer intervention and effort was required to achieve final convergence on all design constraints.

Our characterization experiments performed on the VEX processor show that the contribution of level-shifters to total power is limited to at most 5% of total processor power. Indeed, the considered technology libraries are optimized for low power and for this reason leakage power is not a major concern. However, as mentioned before, the main drawback of AVS is the area overhead. Our experiments indeed show that the impact of level-shifter insertion on logic chip area is around 30%.

Our first objective in REALITY is to privilege post-silicon tuning techniques that can be applied with minimal designer intervention within the synthesis flow. For this reason, we explored ABB as an alternative solution for the final integration in WP5. In the next section we give an overview of ABB techniques, then we discuss an innovative FBB strategy we designed in this WP3 work.

11.3. ABB Techniques

A number of previous works have used ABB for compensating process variations to improve design yield. In [8], 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 do not provide results of applying ABB on complex designs. In [6] 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 [9] 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 macro-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 [10] uses FBB to achieve active leakage power savings. They propose to use high Vth 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 Vth). The authors target to reduce the active leakage power by applying FBB but they do not compensate for timing variability. They also do not 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 [3], proposes to cluster the design. Each cluster is a sub-set of gates in a block, to which optimal body bias is applied to compensate for process variation. The authors 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 [3], for lower number of clusters, the clustering might resulting in dissimilar gates being clustered together resulting in higher leakage power cost. Moreover, physical clustering according to PDFs may lead significant timing penalties. Gate placement heavily influences wiring delays, hence timing-driven placement algorithms should be minimally perturbed to avoid timing violations and poor design convergence.

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, a unit of clustering is a single row and we thereby minimize area overhead due to placement perturbations or adjacent gates having different body bias voltages (we still incur a small area overhead due to well separation required when adjacent rows are biased with different body bias voltages).

12. Physically Clustered FBB Technique

In this section, we describe of our design-time variation 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.

12.1. 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 require 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 [9] 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 [5] 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". Our ABB methodology is compatible with both approaches. However, we believe that in aggressively scaled technologies it may be difficult to ensure strong timing correlation between critical path replicas and the actual critical paths in the circuit, hence we assume that timing-monitoring (also called crystal ball) flip-flops are used for timing sensing.

In principle, if the 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. *We assume that all flip-flops which are end-points of potentially critical paths for a given β are replaced with crystal-ball FFs, and that the global alarm signal for a given data-path is obtained as the OR of the alarms of all crystal-ball FFs.*

12.2. Forward Body Bias (FBB) Technique

FBB is applied to a design with slow gates (hence high V_{th}) to bring back the V_{th} to its target value, or more in general, to achieve gate speedup by lowering the threshold voltage of its transistors . The 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 the STM 45nm CMOS SOI process and the results for a simple inverter is shown in Figure 1.

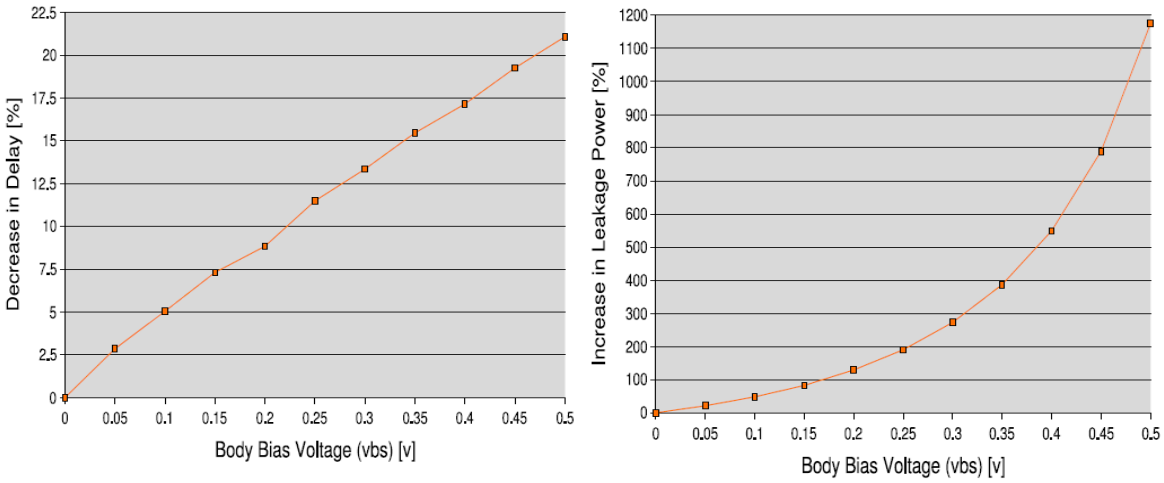

Figure 1: Delay and Leakage power variation of an inverter with varying body bias voltage in 45nm CMOS.

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 Vdd. 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 [11] limits the range of vbs to 0V-0.5V. So we restrict the body bias voltage to these values in all our experiments. FromFigure 1Figure 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.

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 [6], 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 [6]. Here 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 Tci (*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.

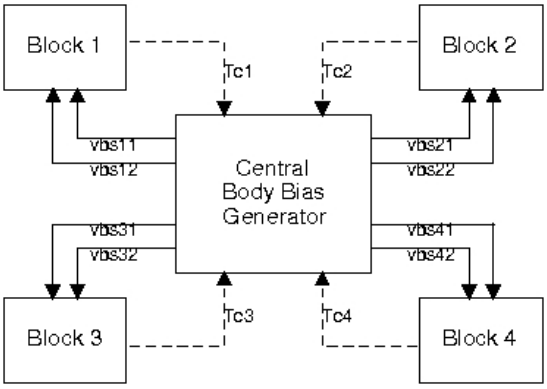

Figure 2: A block diagram showing FBB implementation methodology.

12.3. FBB implementation

In this section, we show how we have implemented FBB on a traditional standard cell based layout style. Figure 2 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.

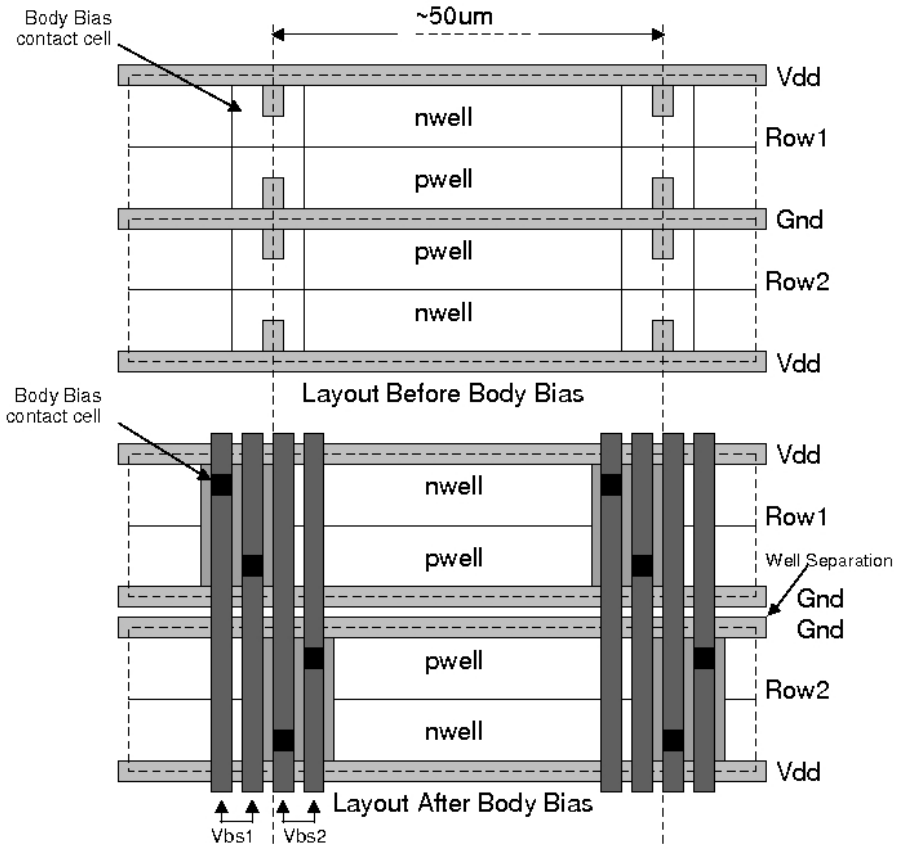

Figure 3: An abstract view of a portion of the standard cell layout with FBB.

12.4. 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.

In this work, we apply only FBB while one can combine both FBB and RBB together to achieve more leakage power savings. However this is a simple extension of the methodology provided here. 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.

12.5. 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 previously explained. 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 i-th row assigned to j-th body bias voltage for i $\in$ 1... N and j $\in$ $1,...,P$. 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 [29], we use the heuristic as described in [29] 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=(p1,...,pM). For each path pi $\in$ Pi, we maintain information on the path delay of that path (pi) and the list of cell instances along that path. We further prune this set to have only those paths which violate the timing i.e.if the degraded path delays (p_d * (1+β)) exceed D_{crit}, the critical path delay of the design. A *timing feasible solution* is one where after we apply FBB, the path delay of any of the path in the path set P_i does not exceed D_{crit}. Note that one can also leverage the available statistical static timing analysis (SSTA) engines to obtain the critical path set.

12.6. 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:

$$\text{Minimize} \sum_{i=1}^{N} \sum_{j=1}^{P} x_{i,j} * L_{i,j} \quad (1)$$

Subject to

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

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

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

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

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 P_i. The value $a_{i,j,k}$ denotes the reduction in path delay of a path $k \in P_i$ 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, $l \in \{1,...,Q_{i,k}\}$ 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 δ_l, $l \in \{1,\ldots,Q_{i,k}\}$. Then,

$$a_{i,j,k} = \sum_{k=1}^{Q_{i,k}} \partial_l$$

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 $k \in \{1,...,M\}$ indicates the speed-up required for each path k in the critical path set Pi. This value is computed as $b_k = D_{crit} - (p_k*(1+\beta))$ for $k = \{1,...,M\}$. Note that there are as many constraints (*M*) as the number of paths in the critical path set Pi. 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 [30] 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.

12.7. Heuristic approach to FBB allocation

In this section, we propose a linear run-time heuristic, to solve the FBB clustering and allocation problem. It is 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 Figure 4. The core of the algorithm is the *CheckTiming* routine 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 Pi *Line1*, We compute and sum the co-efficients $a_{i,j,k}$ for all the rows in the design *Lines 2-7*. 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 obtained 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

$$c_{ti} = \sum_{k=1}^{M} \sum_{k=1}^{Q_{i,k}} \frac{1}{slack_k}$$

$slack_k$ 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 $Tot_{cost} = C_{pass\text{-}one} + C_{pass\text{-}two}$. $C_{pass\text{-}one}$ is a constant number and the cost of $C_{pass\text{-}two} = O(P*N)$. So the total cost $Totcost = O(P*N)$ or $O(N)$. So the run-time of the heuristic algorithm is linear in the number of rows in the design.

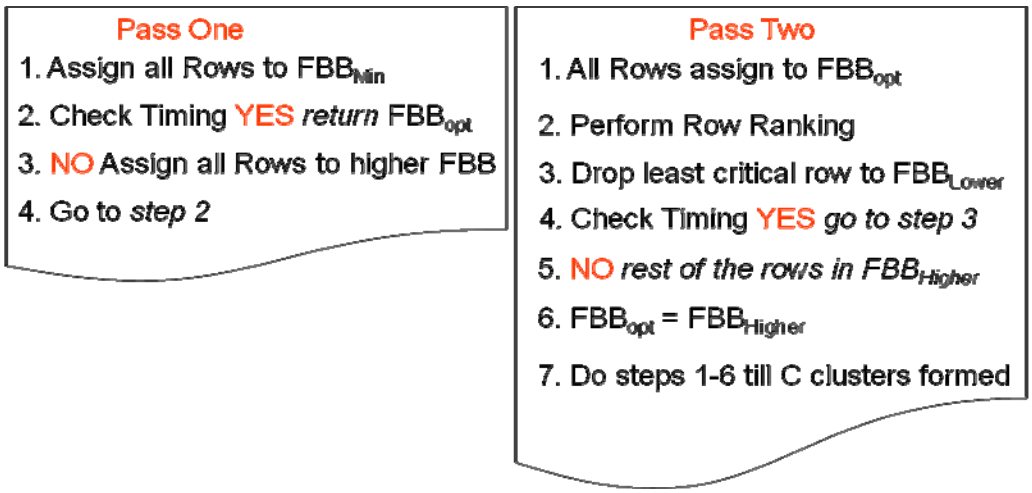

Figure 4: Two Pass Heuristic Clustering Algorithm.

12.8. 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 ISCAS85 and ISCAS89 suites, while the remaining four are large datapaths with controllers belonging to 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 STM 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.

Benchmark	*Gates* [#]	*Rows* [#]	β [%]	*Single BB* [uW]	*ILP* $C=2$ [%]	*ILP* $C=3$ [%]	*Heuristic* $C=2$ [%]	*Heuristic* $C=3$ [%]	*No.Constr*
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

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. *Beta* indicates the circuit slow-down co-efficient. 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, 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 *beta*=5% and 47.6% in case of beta=10%. We see that the savings achieved is higher in case of higher *beta* 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 *β=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 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 [31] 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 few hours. We implemented our row-based FBB on an example design C5315 and the complete placed and routed layout. 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, shown in Figure 5.

Concerning timing constraints, we use a standard strategy for setting them. We first synthesize with zero timing constraint. We get a critical path, then we increase its length by a percentage (15% was used in this case). We perform a synthesis with this value of timing constraint. We have not performed an exploration vs. tightness of timing constraints in this work. We expect this technique to do very well if not all path are fully equalized. If this would be of interest for the project, we will perform an exploration wrt to timing constraints in the context of WP5 with the target benchmarks.

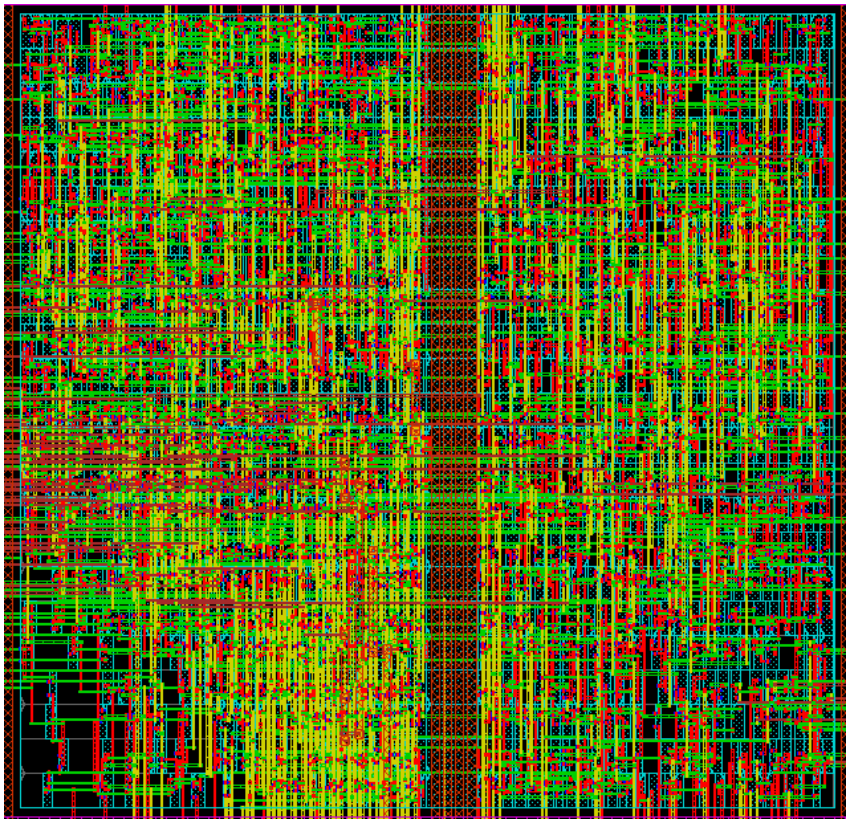

Figure 5: Placed and Routed C5315 design with 2 ***vbs***.

13. Future work in WP3

Future work will focus on a few enhancements to the methodology presented in this deliverable. First, the approach will be tested on a few other larger benchmarks, including processor data-path and controllers. Second, the cost of monitoring circuits and the system controller will be assessed in depth. We do not expect the extra cost to be significant with respect to block-level ABB approaches. On the contrary, our clustering method may also help in reducing the number of paths that should be monitored by crystal-ball FF, thereby further decreasing overhead. We will also improve the integration of our methods within state-of-the-art industrial synthesis flow taking into account detailed information provided by partners STM and ARM. Finally, we will make it possible to read in information on distributions of timing variations from the output of the VAM flow developed by IMEC. With these enhancement, the ABB approach described in this work will become a fully integrated piece of the vertical design flow pursued by the REALITY project. This integrated approach will be applied in WP5 to demonstrate its applicability to industry-relevant benchmarks.

14. References

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

[2] M. Mani, A. Devgan, M. Orshansky, “An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints“, *Proc. Of ACM/IEEE Design Automation Conference*, pp. 309-314, June 2005.

[3] S. Kulkarni, D. Sylvester, D. Blaauw, “Design-Time Optimization of Post-Silicon Tuned Circuits using Adaptive Body Bias“, *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, Vol. 27, No. 3, March 2008, pgs. 481-494.

[4] S. V. Kumar, C. H. Kim , S. S. Sapatnekar, “Mathematically assisted adaptive body bias (ABB) for temperature compensation in gigascale LSI systems“,pp. 559-564, *ASPDAC 2006*.

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

[6] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, V. De, “Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage“, *IEEE Journal of Solid-State Circuits*, pp. 1396-1402, November 2002.

[7] S. Narendra, A. Keshavarzi, B. A. Bloechel, S. Borkar, V. De “Forward body bias for microprocessors in 130-nm technology generation and beyond“ *Solid-State Circuits, IEEE Journal of* Vol. 38, No. 5, May 2003.

[8] C. Neau , K. Roy, “Optimal Body Bias Selection for Leakage Improvement and Process Compensation over Different Technology Generations“, *IEEE International Symposium on Low-Power Electronics and Design*, pp. 116-121, August 2003.

[9] R. Teodorescu, J. Nakano, A. Tiwari, J. Torrellas, “Mitigating Process Variation with Dynamic Fine-Grain Body Biasing“, *International Symposium on Microarchitecture (MICRO)*, December 2007.

[10] 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.

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

[12] E.Humenay, D.Tarjan, K.Skadron, ”Impact of Parameter Variations on Multi-Core Chips”, Int.Workshop on Architectural Support for Gigascale Integration, 2006.

[13] J.Tschanz et al., ”Effectiveness of Adaptive Supply Voltage and Body Bias for Reducing Impact of Parameter Variations in Low Power and High Performance Microprocessors”, IEEE Journal of Solid-State Circuits, vol.38, no.5, May 2003, pp.826-829.

[14] 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 Journal of Solid-State Circuits, vol.37, no. 11, November 2002, pp.1396-1402.

[15] S.Bhunia, S.Mukhopadhyay, K.Roy, ”Process Variations and Process-Tolerant Design”, Int’l Conf. on VLSI Design, pp.699-704, 2007.

[16] H.Chang, S.S.Sapatnekar, ”Statistical Timing Analysis Considering Spatial Correlations Using a Single PERT-like Traversal”, Proc. Int’l Conf. Computer-Aided Design, pp.621-625, 2003.

[17] A.Datta et al., ”Delay Modeling and Statistical Design of Pipelined Circuit under Process Variations”, IEEE TCAD, pp.2427-2436, 2006.

[18] S.Ghosh, S.Bhunia, K.Roy, ”A New Paradigm for Low-Power, Variation-Tolerant Circuit Synthesis Using Critical Path Isolation”, Int’l Conf. on Computer-Aided Design, pp.619-624, 2006.

[19] A.Tiwari, S.R.Sarangi, J.Torrellas, ”ReCycle: Pipeline Adaptation to Tolerate Process Variation”, ISCA 2007, pp.323-334, June 2007.

[20] X.Liang, D.Brooks, ”Mitigating the impact of process variations on CPU register file and execution units”, Int.Symposium on Microarchitecture, December 2006.

[21] D.Ernst et al., ”Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation”, Int. Symp. on Microarchitecture, page 7, Dec. 2003

[22] S.Das, D.Roberts, S.Lee, S.Pant, D.Blaauw, T.Austin, K.Flautner, T.Mudge, ”A Self-Tuning DVS Processor Using Delay-Error Detection and Correction”, IEEE Journal of Solid-State Circuits, vol.41, no. 4, pp.792-804, April 2006.

[23] S.Lee, S.Das, T.Pham, T.Austin, D.Blaauw, T.Mudge, ”Reducing Pipeline Energy Demands with Local DVS and Dynamic Retiming” Int. Symposium on Low Power Electronics and Design, pp.319-324, August 2004.

[24] D.E.Lackey, P.S.Zuchowski, T.R.Bednar, D.W.Stout, S.W.Gould, J.M.Cohn, ”Managing Power and Performance for System-on-Chip Designs Using Voltage Islands”, Int’l Conf. on Computer Aided Design, Nov.2002, pp.195-202.

[25] J.Hu, Y.Shin, N.Dhanwada, R.Marculescu, ”Architecting Voltage Islands in Core-Based System-on-a-Chip Designs”, ISLPED’04, pp.180-185, 2004.

[26] Liangpeng Guo, Yici Cai, Qiang Zhou, Xianlong Hong, ”Logic and Layout Aware Voltage Island Generation for Low Power Design”, EDA Lab, Department of Computer Science and Technology, Tsinghua University, Beijing, China

[27] H.Wu, I.Liu, M.Wong, and Y.Wang, ”Post-Placement Voltage Island Generation under Performance Requirements”, Int’l Conf. on Computer-Aided Design, pp.309-316, 2005.

[28] H.Wu, M.D.F.Wong, ”Improving Voltage Assignment by Outlier Detection and Incremental Placement”, DAC 2007, pp.459-464, 2007.

[29] A. Ramalingam, and et al., ”Sleep transistor sizing using timing criticality and temporal currents”, *ASPDAC-05*, pp. 1094-1097, Jan. 2005.

[30] Frederick S. Hillier, Gerald J. Liberman, “ Introduction to Operations Research,” *Eighth Edition*.

[31] ftp://ftp.es.ele.tue.nl/pub/lp solve.

