Variability-Driven Module Selection with Joint Design Time Optimization and Post-Silicon Tuning

Feng Wang, Xiaoxia Wu, Yuan Xie
The Pennsylvania State University, University Park, PA, USA
{fenwang, xwu, yuanxie}@cse.psu.edu

***Abstract*— Increasing delay and power variation are significant challenges to the designers as technology scales to the deep sub-micron (DSM) regime. Traditional module selection techniques in high level synthesis use worst case delay/power information to perform the optimization, and therefore may be too pessimistic such that extra resources are used to guarantee design requirements. *Parametric yield*, which is defined as the probability of the synthesized hardware meeting the performance/power constraints, can be used to guide design space exploration. The parametric yield can be effectively improved by combining both *design-time* variation-aware optimization and *post silicon tuning* techniques (such as adaptive body biasing (ABB)). In this paper, we propose a module selection algorithm that combines design-time optimization with post-silicon tuning (using ABB) to maximize design yield. A variation-aware module selection algorithm based on efficient performance and power yield gradient computation is developed. The post silicon optimization is formulated as an efficient sequential conic program to determine the optimal body bias distribution, which in turn affects design-time module selection. The experiment results show that significant yield can be achieved compared to traditional worst-case driven module selection technique. To the best of our knowledge, this is the first variability-driven high level synthesis technique that considers post-silicon tuning during design time optimization.** [1]

I. Introduction

Designers have resorted to technology scaling [1] to enhance performance. For example, Intel recently announce that the new Intel processors will be built upon the most advanced 45 nm technology; Embedded processors such as ARM's Cortex-A8 are already manufactured at the leading-edge 65 nm technology. The reliance on deep sub-micron process technologies for their fabrication has brought the concerns of process variations to the forefront. The challenges in fabricating transistors with very small feature sizes have resulted in significant variations in transistor parameters (such as transistor channel length, gate-oxide thickness, and threshold voltage) across identically designed neighboring transistors (this variation is called *within-die variation*) and across different identically designed chips (this variation is called *inter-die variation*). These manufacturing variations can cause significant performance and power variations for an identical hardware design. For example, Intel has shown that a 30% variation in chip frequency and a 20 times variation in chip leakage are observed in 1000 sample chips fabricated in 180nm technology [2]. As technology scales, the performance variations are even more pronounced. It has been predicted that the major design focus for sub-65nm VLSI design will shift to dealing with variability [2].

Traditionally, performance/power variations are handled by a combination of *speed/power binning* and *design margining* [2]: speed/power binning tests all fabricated chips, those of which with a slower speed or excessive power are either discarded or sold at a reduced price; design margining uses worst-case process corners to guarantee the design requirement. However, these solutions are becoming insufficient as the variability increases along with technology scaling, and may not be a viable solution when the variability encountered in the new process technologies becomes very significant. Also, cost sensitivity makes designing for the worst-case manufactured hardware unacceptable.

To bring the process-variation awareness to the design flow, a new metric called *parametric yield* has been introduced [2]–[4]. The parametric yield is defined as the probability of the design meeting a specified constraint $Yield = P(Y \leq Y\_max)$, where Y can be performance or power. To maximize design yield, designers can rely on two complementary strategies: *design-time statistical optimization* and *post-silicon tuning*:

- Design time statistical optimization approaches, such as gate sizing and multiple vdd/vth selection, use statistical timing/power analysis to explore design space, and maximize parametric yield. The design decisions are the same for all fabricated dies and the decisions are made at the design-time (i.e., pre-silicon). As a result, some dies may inevitably miss the target power-delay envelop.
- Post silicon optimization approaches are performed after the fabrication. Techniques such as adaptive body biasing (ABB) and adaptive supply voltage [3], [5]–[7] can be used to tune the fabricated chips, such that the variation in delay/power can be reduced. Compared to design-time solution, the post-silicon tuning decision is different for each differently fabricated die. For example, FBB (Forward Body Biasing) can be applied to slower dies such that the delay becomes faster at the expense of higher leakage power, and RBB (Reverse Body Biasing) can be applied to faster dies such that the circuit is slowed down but the power is reduced.

The majority of the existing analysis and optimization techniques related to process variations are at the lower level (device or logic gate level). In the domain of high-level synthesis, process-variation-aware research is still in its infancy [4] [8]. *It is important to raise the process variation awareness to a higher level*, because the benefits from higher-level optimization often far exceed those obtained through lower-level optimization. Furthermore, higher-level statistical analysis enables early design decisions to take lower-level process variations into account, avoiding late surprise and possibly expensive design iterations.

In this work, we propose a variability-driven module selection algorithm that combines design-time optimization with post-silicon tuning (using adaptive body biasing) to maximize design yield. To the best of our knowledge, this is the first variability-driven high level synthesis technique that *considers post-silicon tuning during design time optimization*.

II. Related Work

High level synthesis (HLS) is a well-studied problem [9] [10] [11]–[15] and we have seen many successes in industrial practice. For example, the Mentor Graphics Catapult C high-level synthesis tool [16]

[1]This research was supported in part by NSF grants of CAREER 0643902, CNS 0720659 and CCF 0702617.

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

has been used by many companies in the design flow. Early high-level synthesis mainly focused on performance and cost trade-offs [9] [10]. Then researchers proposed low-power HLS techniques [11]–[13], as well as fault-tolerant HLS techniques [14], to reduce power consumption and improve reliability, respectively. Recently, thermal-aware HLS techniques [15] were also proposed to mitigate the temperature increase due to higher power density as technology scales. These existing high level synthesis methodologies explore the tradeoffs among performance, cost (area), and/or power/reliability. However, the design space exploration has been limited to deterministic optimization using either constant delay/power values or worst-case delay/power values for the underlying hardware. These approaches are blind to the impact of the process variations on delay or power of the hardware design, and may result in overly pessimistic synthesis results.

Although the process variation problem was well recognized more than a decade ago [17], it did not become a major research focus until recent years [18]. Early statistical timing analysis approaches were based on Monte Carlo techniques [17], which are expensive in terms of computation complexity and therefore are not suitable for large circuits. Recently, many fast and efficient gate-level state-of-the-art statistical timing analysis methods have been proposed [19]. In sub-90 nm technologies, the major component of the total power consumption will be leakage power, which exhibits large variations under the influence of process variations. As a result, gate-level statistical power analysis methods [18] have been developed. Based on the statistical timing analysis and power analysis, various statistical optimization approaches have been proposed, such as statistical minimization of the total power under timing yield constraints [18], statistical gate sizing to improve the probability of meeting timing constraints [20]. Recently, Marculescu et al. developed a statistical performance analysis approach for the embedded system [21] [22].

In the domain of high-level synthesis, process-variation-aware research is still in its infancy. Recently, Hung et al. [4] use performance yield (defined as the probability of the synthesized hardware can meet performance requirement) to guide a Simulated-Annealing based HLS framework. However, each of the synthesis steps is still deterministic oriented, and power variation for function units is not considered. Mohanty et al.'s work [8] takes into account the leakage power variation in high level synthesis, but time consuming Monte Carlo techniques is a key limiting factor of this design time technique. Jung et al. [23] proposed a timing variation aware scheduling and resource binding algorithm to improve the latency.

In this paper, a two-stage optimization approach in module selection is proposed to *take into account the post-silicon tuning at the design-time optimization* in high level synthesis. At the first stage, an iterative variation aware module selection algorithm is employed considering both power and performance variability. At the second stage, a sequential second order conic programming algorithm is employed to determine the optimal body bias for post-silicon tuning, which in turn affects design-time module selection.

III. Preliminaries and Problem Formulation

A. The Influence of Process Variation on HLS

In HLS, each operation (such as addition and multiplication) in a control data flow graph (CDFG) is scheduled to one or more cycles (or control steps). Each control step corresponds to a time interval equal to the clock period. Each operation may be performed by more than one *compatible* resource type from the resource library. For example, the addition operation can be performed by either a ripple-carry adder or a carry look-ahead adder, which have different delay, power, and area parameters. *Module selection* decides the type of functional units to perform the operations in the CDFG. The same resource (functional units or registers) can be shared to perform multiple operations or store more than one variable. Traditionally, high-level synthesis is performed under resource constraints and performance constraints. Resource constraints require that the operations are performed with only a limited number of resources available; performance constraints require that the operations in the CDFG finish execution in a number of clock cycles with a particular clock rate (clock cycle time). Note that in this paper, we focus on *data-flow intensive* applications, in which most of the computations performed in the design are arithmetic operations (such as addition and multiplication), even though the approach could be easily extended for *control-flow intensive* applications, which contain significant control-flow constructs such as nested loops and conditionals.

Traditionally, worst-case delay/power parameters for the resource are used to facilitate the module selection. However, it is becoming inappropriate as larger variability is encountered in the new process technologies. For example, Fig. 1 shows the delay variations (depicted as normalized sigma/mean) for 11 different type of 16-bit adders that span a range of circuit architecture and logic evaluation styles, all of which are implemented in IBM Cu-08 (90nm) technology [24].

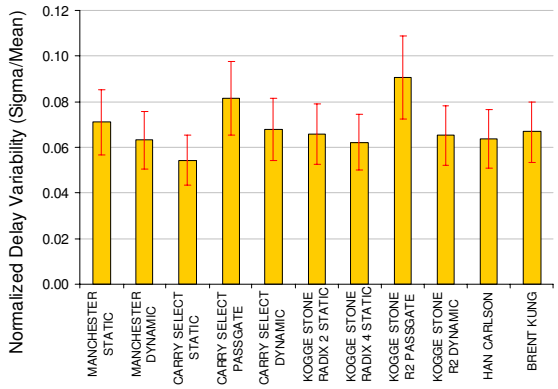


Fig. 1. The delay variation (normalized sigma/mean) for 16-bit adders in IBM Cu-08(90nm) technology (Courtesy of K. Bernstein, IBM [24]).

Due to the large variation in delay and power, the existing deterministic worst-case design methodologies in HLS may result in unexpected performance discrepancy or a pessimistic performance/power estimation, or may end up using excess resources to guarantee design constraints, due to overly conservative design approaches.

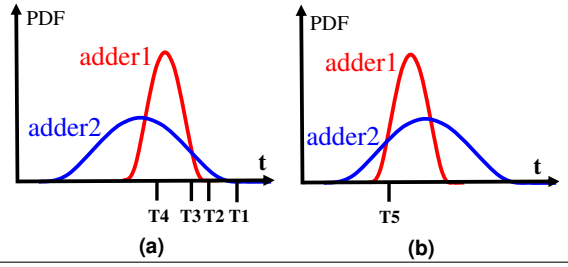
Furthermore, worst-case analysis without taking the probabilistic information into account can also result in a pessimistic estimation. For example, assume that the delay of an adder ($X = D_{add}$) and a multiplier ($Y = D_{mul}$) have independent Gaussian distribution $N(\mu, \sigma^2)$. In conventional worst-case analysis, the worst-case execution time (WCET) is calculated as $\mu + 3\sigma$ (3σ delay). Based on the statistical information, $X+Y$ follows ($N(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$), and the WCET for $X+Y$ is $WCET_{X+Y} = \mu_X + \mu_Y + 3\sqrt{\sigma_X^2 + \sigma_Y^2}$, which is smaller than the sum of the WCET of each function unit ($WCET_X + WCET_Y = \mu_X + \mu_Y + 3(\sigma_x + \sigma_y)$). Therefore, simply adding the WCET of two function units can result in a pessimistic estimation of the total delay, and may end up using excess resources to guarantee performance constraints.

B. Variation-aware Module Selection

Similar to gate-level optimization, we bring the process-variation awareness to the high-level synthesis flow with the *parametric*

yield concept. The *performance yield* is defined as the probability of the synthesis results meeting the clock cycle time constraints under the latency constraints and resource constraints. The *power yield* is defined as the probability that the total power of the synthesis result is less than the power limit under latency and resource constraints. The performance and power yield analysis methods will be described in Section V and VI.

It is obvious that the parametric yield of the HLS resultant hardware depends on all steps of high-level synthesis: scheduling, module selection, resource sharing, and clock selection. These steps are usually interacted with each other during high-level synthesis, and influence the final parametric yield calculations. However, due to the space limitation, this paper will focus on the module selection problem, assuming other synthesis tasks have been performed. The variation aware module selection is formulated as: *Given a scheduled data flow graph, with the latency and resource constraints, and a resource library with statistical delay and power characterization, determine the type of function units to perform the operations in CDFG, to maximize the power yield subject to performance yield constraints.*



CCT	WCET based	Performance yield based
T1	Adder 2 (WCET $\leq$ CCT, smaller area)	Adder 2 (100% yield, smaller area)
T2	Adder 1 (WCET $\leq$ CCT)	Adder 1 (100% yield)
T3	None	Adder 1 (better yield)
T4	None	Adder 2 (better yield)
T5	None	Adder 2 (better yield)

Fig. 2. An example of module selection for an adder and the comparison of worst-case execution time (WCET) based and performance yield based module selection. The delay distribution of two different type of adders are shown as PDF (probability distribution function), and the area of adder 2 is smaller.

In process variation aware module selection, we take into account the distributions of the delay and the power for each resource type in the library. Fig. 2 shows the complexity of module selection problem even for a simple example, without considering power variation. The example also illustrates the difference between conventional worst-case based module selection and variation-aware module selection. Assume that the synthesis result is a single adder, and there are two types of adders with the delay distribution available in the resource library. In conventional module selection, the worst-case execution time analysis shows that *adder 1* is faster than *adder 2*. When the clock cycle time (CCT) constraint is large (e.g., $CCT = T1$), both adders meet timing constraint and the one with smaller area (adder 2) is selected; When we decrease the the CCT, WCET-based module selection has only one choice (adder 1) or no solution (when both adders' $WCET$ are larger than CCT), while performance yield based approach may select either adder 1 or adder 2, depending on the performance yield (*i.e.*, the probability of meeting the CCT constraint, which could be smaller than 100%).

The most interesting observation from Fig. 2 is that, the performance-yield based selection could be counter-intuitive. For example, a heuristic of using the product of sigma and mean ($\sigma \times \mu$) was proposed [4] to help module selection; however, Fig. 2(a) (when $CCT = T3$ and $CCT = T4$) shows that the heuristic may not be appropriate. Another example is illustrated in Fig. 2(b): adder 1 has smaller mean (μ) and smaller variation (σ), which means it is faster and resistant to delay variation. Therefore, intuitively, it should be a better choice than adder 2. However, if $CCT = T5$, adder 2 is actually a better choice since it has higher performance yield. Note that the example in Fig. 2 does not take into account power variation and is only for a single adder module selection.

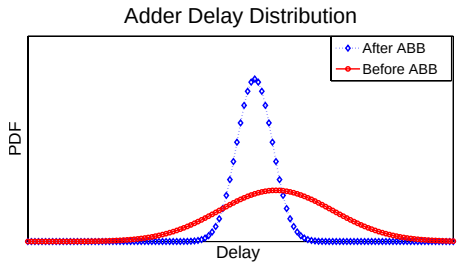


Fig. 3. The adder delay distribution can be adjusted by post-silicon ABB techniques

C. Adaptive Body Biasing

Post silicon tuning techniques can be applied to the fabricated dies after the fabrication. Adaptive Body Biasing (ABB) [3], [6], [25] is an effective technique to reduce the impact of the process variations by controlling the threshold voltage. With bidirectional adaptive body bias, the applied voltage can either raise the threshold voltage of the die (i.e., Reverse Body Biasing (RBB)), to reduce the leakage power at the expense of slowing down circuits, or lower the threshold voltage (i.e., Forward Body Biasing (FBB)), to increase the clock frequency at the expense of higher leakage power. ABB techniques can effectively tighten distribution of the performance and power, thus, the yield loss due to process variation can be minimized. For example, Figure 3 shows that by applying ABB techniques, the delay distribution can be adjusted and the spreading is narrowed. Note that the body bias V_{SB} for each individual die is different, and therefore V_{SB} is a probability distribution derived from the probability distribution of performance/power.

IV. Function Unit Delay and Power Modeling

In this section, the piecewise linearized delay model for function units is first introduced, and the exponential power model of function units is then presented. All these models are simple extension from the gate level models [26] and [18].

In the delay modeling, the delay of the function unit is expressed in terms of the gate length (l), the threshold voltage (V_{th}) and the body bias voltage (V_{SB}). Piecewise linear approximation of the delay has been widely used in the gate level timing analysis. Thus, the delay of the function unit can also be expressed in a piecewise linear function. Suppose ΔV_{th} represents the deviation of the threshold voltage, Δl represents the deviation of the gate length, and V_{SB} represents the applied body biasing, the delay of a function unit, T_i, is expressed as:

$$T_i = a0_i + a1_i \Delta V_{th} + a2_i \Delta l + a3_i V_{SB} \quad (1)$$

where $a0_i$ is the nominal delay computed at the nominal values of the process parameters without body biasing. $a1_i$, $a2_i$ and $a3_i$ represent the sensitivity to the deviation of threshold voltage and gate length, and applied body bias, respectively.

The power consumption of a function unit consists of dynamic power and leakage power. The dynamic power is relatively immune to process variation, while the leakage power is affected by process variation greatly, and it becomes a dominant factor in total power consumption as technology scales to nanometer region [18]. Our statistical leakage power model is based on the gate level model and the rms error of this gate level model is around 8% [18]. In this approach, the leakage power of each logic gate is expressed as a lognormal random variable in a canonical form, and the leakage power dissipation of a function unit, which consists of many gates, can be computed as the sum of these random variables. This sum can be accurately approximated as a lognormal random variable using an extension of Wilkinson's method [18]. Consequently, the leakage power dissipation of a function unit can also be expressed as a lognormal random variable in a canonical form. Therefore, the leakage power of a function unit can be expressed as

$$P_i = exp(b0_i + b1_i \Delta V_{th} + b2_i \Delta l + b3_i V_{SB}) \quad (2)$$

where $exp(b_0)$ is the nominal leakage power computed at the nominal values of the process parameters. b_i are the sensitivities to their corresponding sources of the deviation and the bias voltage.

V. Statistical Analysis For DFG

In this section, we briefly describe our statistical timing/power analysis for a synthesized DFG [13] (in which all operations have been scheduled and bound to module instances selected from the resource library). The terminology and approach is similar to most of the gate-level statistical timing/power analysis approaches [17]–[20], [27]. Although the fundamental idea is the same, that is, to consider process variations during timing/power analysis, the divergence occurs in that the allocated resource can be shared and that the sequencing order of operations with respect to clock cycle time must be enforced in HLS. This divergence makes statistical analysis at high-level synthesis a unique problem. In addition, we introduce parametric yield computation method for a synthesized DFG and present fast yield gradient computation methods.

A. Statistical Timing Analysis in HLS

In the statistical timing analysis for a synthesized DFG, the timing quantity is computed by using two atomic functions sum and max. Assume that there are three timing quantities, A, B, and C, which are random variables. The sum operation $C = sum(A, B)$ and the max operation $C = max(A, B)$ will be developed:

1) The sum operation is easy to perform. For example, if A and B both follow a Gaussian distribution, the distribution of $C = sum(A, B)$ would follow a Gaussian distribution with a mean of $\mu_A + \mu_B$ and a variance of $\sqrt{\sigma_a^2 + \sigma_b^2 - 2\rho\sigma_a\sigma_b}$, ρ is correlation coefficient.
2) The max operation is quite complex. Tightness probability [27] and moment matching [28] techniques could be used to determine the corresponding sensitivities to the process parameters. Given two random variables, A and B, tightness probability of random variable A is defined as the probability of A being larger than B. An analytical equation in [28] to compute the tightness probability is used to facilitate the calculation of max operation.

The delay distribution of module instances can be obtained through gate-level statistical timing analysis tools [19] [27] or Monte Carlo analysis in HSPICE. With the atomic operations defined, the timing analysis for the synthesized DFG can be conducted using PERT-like traversal [19].

B. Statistical Power Analysis in HLS

Our statistical leakage power analysis method is based on the gate level analysis approach [18]. In this approach, the leakage power of each logic gate is expressed as a lognormal random variable in a canonical form, the total power of the circuit can be computed as the sum of these random variables. This sum can be accurately approximated as a lognormal random variable using an extension of Wilkinson's method [18]. Since the leakage power dissipation of a function unit can also be expressed as a lognormal random variable in a canonical form as show in Section IV, the total power dissipation of the synthesized DFG is computed as the sum of the leakage power dissipation of the module instances in the DFG. Thus, this sum can also be approximated as a lognormal random variable in a canonical form using the extended Wilkinson's method. Therefore, the leakage power of each module instance can be expressed as

$$P_m = exp(m_0 + \sum_{i=1}^{n} m_i Y_i + m_{n+1} R_m) \quad (3)$$

where m_0 is the nominal value computed at the nominal values of the process parameters. Y_i represents the correlated variation, and R_m represents the independent random variation. Y_i and R_m are independent and normally distributed random variables with zero mean and unit variance. m_i and m_{n+1} are the sensitivities to their corresponding sources of the variation.

The sum of the power dissipation of two modules is approximated as a lognormal random variable in the same format as expression (3). Assuming that $P_m = P_k + P_n$, the coefficient of P_m can be determined by moment matching [29],

$$m_i = log(\frac{E(P_k e^{Y_i}) + E(P_n e^{Y_i})}{(E(P_k) + E(P_n))E(e^{Y_i})}) \ \forall i \in [1, n] \quad (4)$$

$$m_0 = 0.5 log(\frac{(E(P_k) + E(P_n))^4}{(E(P_k) + E(P_n))^2 + Var(P_k) + Var(P_n) + 2Cov(P_k, P_n)}) \quad (5)$$

$$m_{n+1} = [log(1 + \frac{Var(P_k) + Var(P_n) + 2Cov(P_k, P_n)}{(E(P_k) + E(P_n))^2}) - \sum_{i=1}^{n} m_i^2]^{0.5} \quad (6)$$

where $E(P)$ represents the mean of the random variable P, the $Var(P)$ represents the variance of the random variable P and $Cov(P, Q)$ is the covariance of the random variables P and Q.

C. Statistical Performance Yield Analysis for DFG

In a synthesized DFG, the operations are distributed to the clock cycles and bound to module instances selected from resource library. The operations in each clock cycle must finish execution within that clock cycle. The performance yield is calculated as the probability of the operations scheduled in each clock cycle meeting the clock cycle time constraints under the conditions that latency constraints and resource constraints are not violated. Assuming that the clock cycle time is T_clock, the latency constraints are N clock cycles, and the critical path delay of the operations scheduled in clock cycle i is $Tmax_i$, the performance yield can be computed as

$$Yield_{delay}(DFG) = Prob(Tmax \leq T\_clock | constraints) \quad (7)$$

where $Tmax = max(Tmax_i), \forall i \in [1, N]$. $Tmax_i$ can be computed using the statistical timing analysis described in Section V-A. Note that the max operation is defined in Section V-A. The *constraints* represent the latency constraints and the resource constraints.

D. Performance Yield Gradient Computation for Module Selection

Based on the yield analysis method in the previous sub-section, a yield gradient method is described in this sub-section. A brute-force approach to performance yield gradient computation requires computing the performance yield of the entire synthesized DFG twice. To facilitate the yield computation in the module selection, we employ a *divide and conquer* method to avoid the yield computation over all the clock cycles in a synthesized DFG. The synthesized DFG is divided into blocks and each block contains minimum number of clock cycles (time steps) such that resource shared operations or operations bound to multiple-clock-cycle modules are in the same block. For example, as shown in Fig. 4, the multiplication operation is bound to a two-clock-cycle module and two addition operations share the same module. The $block1$ consists of two clock cycles, $CC2$ and $CC3$, such that the complete two-clock-cycle multiplication operation/two addition operations are in the same block. Assuming that the correlation between the different module instances is relatively small compared to the resource shared and multiple clock cycle operations, we can compute the yield of each block separately and approximate the performance yield of the entire DFG as

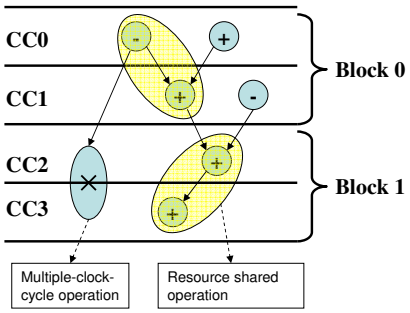


Fig. 4. Yield computation for a synthesized DFG. The multiplication operation is bound to a two-clock-cycle module and two additions share the same module.

$$Yield_{delay} = \prod_{i=1}^{M} Yield_{delay}(b_i) \tag{8}$$

where $Yield_{delay}(b_i)$ is the yield value of $block\ i$ and it can be computed as $Prob(Tmax\_block_i \leq T\_clock|constraints)$, and M is the total number of blocks in the DFG. Thus, assuming that an operation in block j is rebound to a new module, the performance yield gradient of the module change can be computed as

$$\Delta Yield_{delay} = \prod_{i=1, i\neq j}^{M} Yield_{delay}(b_i) \times \Delta Yield_{delay}(b_j) \tag{9}$$

Thus the yield gradient computation for the entire DFG is reduced to the yield gradient computation for a single block in the DFG.

E. Power Yield Gradient Computation for Module Selection

The statistical power computation is performed by summing the power dissipation of each module instance in the synthesized DFG as described in Section V-B. To perform power yield gradient analysis for an operation rebinding, we first perform statistical power analysis of the DFG after the rebinding:

$$P^{new}_{DFG} = P^{old}_{DFG} - P^{old}_{opt_k} + P^{new}_{opt_k} \tag{10}$$

where P^{new} and P^{old} refer to the power dissipation distribution after rebinding and before rebinding, respectively; P_{DFG} and P_{opt_k} denote the total power dissipation of synthesized DFG and the power of module instance bound to operation k, respectively. With the distributions of the power dissipation before rebinding and after rebinding determined, the power yield gradient can be computed as

$$\Delta Yield = Yield(P^{new}_{DFG}) - Yield(P^{old}_{DFG}) \tag{11}$$

where $Yield(P)$ is computed as the probability of P less than the power limit.

VI. Module Selection with Design-Time Optimization and Post-Silicon Tuning

In this section, we first introduce a design time module selection algorithm based on the fast yield computation method. We then present the algorithm on how to decide optimal body biasing, and describe the module selection strategy of joint design time optimization and post silicon tuning. Note that a single cycle operation cannot be bound to a multi-cycle module in module selection. In high-level synthesis, scheduling and module selection interact with each other. Therefore, at each scheduling step, the module selection algorithm will be called to find the optimal instance from the library. Due to the space limitation, we only present our module selection algorithm, with the assumption that the schedule step has been performed.

A. Design Time Variation-aware Module Selection Algorithm

```
Optimization (ISDFG,constraints,Library){
1.  While (ΔYield > ε and meet constraints){
2.    Generate_multiple_moves generates the to_move_list list;
3.    Find k of to_move_list to maximizing the total gain G_k;
4.    If (total gain G_k > 0){
5.      Apply this sequence of moves;
6.      Evaluate the power and performance yield;
7.    }}}

Generate_multiple_moves (ISDFG,Library, constraints){
8.  While (maximum number of moves is not reached) {
9.    For (each possible move in the DFG){
10.     Evaluate the gain of that move;
11.     Save the move and gain to temp_move_list;}
12.   Insert the move with highest gain to to_move_list;
13.   }
```

Fig. 5. The Pseudo Code of Variation Aware Optimization in Module Selection

Our module selection takes an initial scheduled DFG ($ISDFG$), $constraints$ (latency constraints, resource constraints, CCT constraints, and power constraints), and a module $Library$ as inputs, and output a synthesized DFG that is power optimized while satisfying performance constraints. In the initial scheduled DFG, the operations are bound to the fastest module in the module library, which meet the latency constraints in terms of the number of clock cycles. The iterative module selection algorithm consists of two steps: 1) performance yield maximization; 2) power yield improvement under the performance yield constraint.

Note that the optimization algorithm shown in Fig. 5 can be configured as performance optimization or power optimization depending on the *gain* function, and the gain is the change in total yield that results from the move (module selection) : for performance optimization, the gain is the performance yield gain, $\Delta Yield_{delay}$; for power optimization under the performance yield constraint (e.g., the probability of the synthesis result can run at 200 Mhz should be at least 90%), the gain is $\alpha * \Delta Yield_{delay} + \Delta Yield_{power}$, where α is weight factor.

Similar to [30], our iterative variation aware module selection algorithm is based on a variable depth search method. The algorithm starts with an initial scheduled DFG. It identifies the move with the maximum gain, and inserts that move to to_move_list (Line 9-12). The algorithm continues to identify the moves that give max gain until the maximum number of moves is reached (Line 8-12). After all these moves identified, we find a sequence of moves, which gives the best gain (Line 3). In other words, the algorithm find k consecutive moves,

which give best $G_k = \sum_{i=1}^{i=k} g_i$. Note that the gain of some moves might be negative. The algorithm accepts these moves when $G_k > 0$. This makes our algorithm capable of escaping the local minimum. The k consecutive moves are then committed, and the performance yield or power yield are evaluated (Line 5-6). The iteration continues until the yield improvement is less than a preset small value or the delay/resource constraint is violated(Line 1).

B. Post-silicon Tuning with ABB

Once the module selection is decided at the design time, adaptive body biasing (ABB), which is a post-silicon tuning technique, can be applied to further reduce the parametric yield loss. ABB body biasing can be applied at the chip level (i.e., all the modules on a die will have a single body biasing V_{SB}), and the whole chip is either applied FBB or RBB; ABB can also be applied at the module level (i.e., each module can have its own V_{SB}), such that each module can be applied FBB or RBB, achieving a finer granularity of tuning. After fabrication, the optimal body biasing for each die can be determined by speed/power binning. However, at design time, the optimal body biasing V_{SB} (either chip-level or module-level) is a probability distribution, depending on the statistical delay/power distribution of the chip.

In this section, we describe how to decide the optimal body biasing for a particular module selection decision, such that the power yield is maximized under the performance constraints (i.e., meeting a particular performance yield requirement). The optimization can be formulated as a sequential of a second order conic program problem [31]. To obtain the optimal V_{SB} to compensate the variability caused by the process variations, we formulate the optimization problem as follows:

$$minimize\ P_{sttot} \tag{12}$$

$$subject\ to\ P(Tmax < T_{clock}|constraints) > \alpha \tag{13}$$

where the objective function P_{sttot} is defined as $\bar{P}_{tot} + \beta * \sigma_{P_{tot}}$. $\bar{P}_{tot}$ and $\sigma_{P_{totl}}$ are the nominal and the variance of the total power, respectively. β is a weighting factor, which balances the optimization effort on reducing the mean and the spread of the power distribution. α is the required performance yield. The *constraints* represent the latency constraints and the resource constraints. $Tmax = max(Tmax_i), \forall i \in [1, N]$ as shown in equation (7).

In the optimization problem, the applied body biasing V_{SB}, is the optimization variable to be determined. Applied body bias is used to compensate the process variation. Thus, V_{SB} is a function of the random variables. In Mani et al.'s work [3], V_{SB} is set to be an affine function of the parameter variations. To simplify the optimization problem, which can be solved using a conic programming, we set

$$V_{SB} = s^T Y \tag{14}$$

and the vector s is the compensation vector. Thus the vector s is the optimization vector to be determined. Y is the random variable vector used to model process variations, which consists of the random variables Y_i and R_m in equation (3).

The constraint function of this optimization problem is the integration of the delay distribution, $Tmax$. This random variable, $Tmax$, is obtained by performing max operation over the timing quantities, T_i. Thus, the constraint functions need to be transformed. First, since the V_{SB} is an affine function of the variation parameters, the delay of a module instance can be expressed as a linear function of optimization variables:

$$T_i = h_i^T s \tag{15}$$

Second, to compute $Tmax$, the max operation over the timing quantities T_i is performed. The max operation is complex and involves the integration of the exponential function. Thus, the max operation has to be approximated. According to Clark's work [28], the max operation can be computed as a linear function under specific constraints as follows:

Assuming that $Tmax = max(Tmax_0, ..., Tmax_n)$, we then have

$$Tmax = \Sigma_{i=1}^{n} C_i Tmax_i \tag{16}$$

where C_i is the probability of the timing quantity, $Tmax_i$, determining the $Tmax$. To make this approximation valid, a new constraint is introduced to limit the change of body bias V_{SB} to be less than a small value, ϵ. In this paper, the change of the body biasing, ΔV_{SB}, is set to be 0.01. From equation (15) and (16), we can express $Tmax$ as a linear combination of the optimization vector s,that is

$$Tmax = b^T s \tag{17}$$

Given that $Tmax$ is a gaussian random variable, the constraint, $P(Tmax < T_{limit}) < \alpha$, can be transformed to a quadratic function [31]:

$$\bar{T}max + \phi^{-1}(\alpha)(\sigma_{Tmax}) \leq T_{limit} \tag{18}$$

where $\bar{T}max$ and σ_{Tmax} are the mean and variance of $Tmax$. Thus, we can express the constraint function in terms of optimization vector s,

$$\bar{b}^T s + \phi^{-1}(\alpha)(s^T \Sigma s)^{1/2} \leq T_{limit} \tag{19}$$

The objective function, P_{sttot}, takes into account the mean and variation of the power. However, the objective function is an exponential function of random variables. To efficiently solve the optimization problem using second order conic programming, the objective function is transformed. According to probabilities theory, the mean and variance of the log-normal random variable can be expressed as a exponential function. Consequently, the objective function is a complex nonlinear function of the optimization vector s. Thus, Taylor expansion is used to obtain its linear approximation. We then have:

$$\bar{P}_{DFG} = P_{DFG}(s_{ini}) + (s - s_{ini})^T \nabla_s (\bar{P}_{DFG}) \tag{20}$$

$$\sigma_{P_{DFG}} = \sigma_{P_{DFG}}(s_{ini}) + (s - s_{ini})^T \nabla_s (\sigma_{P_{DFG}}) \tag{21}$$

where s_{ini} represents the initial value of the optimization variable vector. To simplify the notation, $a1$ and $a2$ is set to be $\nabla_s(\bar{P}_{DFG})$ and $\nabla_s(\sigma_{P_{DFG}})$ respectively. We then express the objective function as a linear function of the optimization vector s:

$$a1^T s + \beta * a2^T s \tag{22}$$

Based on the above transformations in equations (22) and (19), the optimal body biasing assignment problem can be formulated as a second order conic program:

$$minimize \quad (a1 + \beta * a2)^T s \tag{23}$$

$$subject\ to \quad \bar{b}^T s + \phi^{-1}(\alpha)(s^T \Sigma s)^{1/2} \leq T_{limit} \tag{24}$$

$$c^T (s - s_{ini}) < \epsilon \tag{25}$$

The linear constraint is set to make the approximation of $Tmax$ valid.

Based on this second order conic program formulation, the sequential conic program (SCP) algorithm is shown in Fig. 6. In the sequential programming algorithm, the complex optimization problem is transformed to a sequence of the simplified subproblem. In this work, the subproblem, CP^i, is used as an approximation of

the optimal body biasing problem in a range, ϵ. In this work, the subproblem is solved using the above second order conic program. At the end of each iteration, the solution of the subproblem, CP^i, is evaluated for convergence. The iteration exits when the convergence test fails, or the number of maximal tries has been reached.

SCP ($ISDFG$,$constraints$,s)
1. While (convergent){
2. setup the $CP^i(\epsilon)$
3. solve the $CP^i(\epsilon)$
4. }

Fig. 6. The Pseudo code of optimal body biasing of DFG

C. Joint Optimization Algorithm

JointOpt ($ISDFG$,$constraints$,$Library$)
1. While ($\Delta Yield > \epsilon$ and meet $constraints$){
2. *Design time module selection under current body bias*;
3. *Sequential Conic Optimization*;
4. }

Fig. 7. The Pseduo Code of Variation Aware Optimization of DFG

Our joint design time module selection and post silicon tuning algorithm takes an initial scheduled DFG, $constraints$ (latency constraint, resource constraint, CCT constraint, and power constraint), and a module $Library$ as inputs, and outputs a synthesized DFG with optimal body bias that is power optimized while satisfying performance constraints. In the initial scheduled DFG, the operations are bound to the fastest module in the module library, which meet the latency constraints in terms of the number of clock cycles.

Our joint optimization algorithm consists of two steps: 1) design time module selection algorithm selects the module instance for the operations in DFG to maximize the power yield under power yield constraints, and the module selection is performed under the body bias determined in previous iteration; 2) the sequential conic program determines the optimal body bias for the module instances in the current iteration to tighten the delay and power distribution to further improve the power yield. The body bias is initially set to be zero. These two steps are iterated until no improvement can be obtained.

VII. Analysis results

In this section, we present the analysis results and show that our joint design time and post silicon time method can effectively reduce the impact of the process variation and maximize the parametric yield, as compared to traditional worst-case based design-time module selection method.

We implement our joint design time module selection and post silicon tuning algorithm in C++ and conduct the experiments on six high level synthesis benchmarks: a 16-point symmetric FIR filter (FF), a 16-point elliptic wave filter (EWF), an autoregressive lattice filter (ARF), an algorithm for computing Discrete Cosine Transform (DCT), a differential equation solver (DES), and an IIR filter (IIR). The resource library contains different adders and multipliers, implemented in 90 nm technology, with statistical delay and power distributions, as shown in Fig. 1. The library characterization is performed based on the results of Monte Carlo analysis in HSPICE. A few experiments are conducted to demonstrate the effectiveness of our algorithm, with the average runtime of algorithm for all benchmarks (on a 2GHz Pentium 4 Linux machine) less than 0.5 sec:

- *Design-time only variation-aware vs. Worst-case deterministic module selection.* First, we compare our variation-aware design time optimization approach (*without considering post-silicon tuning*) to the traditional worst case deterministic module selection method. The power optimization is performed under 90% performance yield constraints. As shown in Fig. 8, the reference design time approach has significant yield improvement over the worst case design time approach. An average 34% yield gain is achieved using our reference variation design time approach.

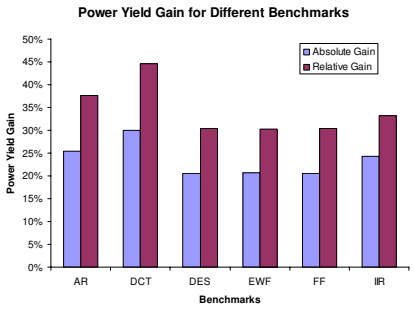


Fig. 8. Power yield improvement over worst-case based deterministic module selection, with 90% performance yield constraint.

- *Joint design-time and post-silicon tuning Vs. design-time only variation-aware module selection.* We then compare the results of our joint design-time and post-silicon tuning module selection algorithm (JT) against the variation-aware design-time only module selection method (DT). Table I shows the results of our method with a single chip-level body bias control (JTS) against those of the design time module selection technique (DT), under a 99% performance yield requirement. From the second column to the third column, we show the absolute power yield results of the joint optimization method (JTS) and the design-time only module selection technique (DT), respectively. In the fourth column, we show the absolute value of the yield improvement of our method over design-time only method. In the fifth column, we show the relative yield improvement of our method over the design time method. As we can see from Table I, significant yield improvement could be obtained if we take into account post-silicon tuning techniques in high level synthesis. The yield results show that joint design time and post silicon optimization can achieve average 38% power yield improvement, compared to design-time only variation-aware module selection. If we relax the power constraint, we may have larger power yield. For example, Table II shows that for the same 99% performance yield constraint, if we relax the power constraint, joint design time and post silicon optimization can achieve average 11% power yield improvement.

TABLE I
Power Yield Under 99% Performance Yield Constraint

Name	DT	JTS	JTS-DT	(JTS-DT)/DT
AR	47%	86%	39%	83%
DCT	60%	85%	25%	42%
DES	76%	90%	14%	18%
EWF	79%	90%	11%	14%
FF	75%	92%	17%	23%
IIR	58%	85%	27%	47%
Average	66%	88%	22%	38%

- *Module-level post-silicon tuning*. The more aggressive approach is module-level post-silicon tuning (i.e., each module can have its own V_{SB} control). This approach results in a finer granularity tuning, and can effectively address the *intra-die variation*). However, the additional overheads of the body bias generator and control circuitry can adversely affect the gain of this approach. Therefore, it is appropriate to apply post-silicon tuning at multiple-module level instead of single module level.

TABLE II
POWER YIELD UNDER 99% PERFORMANCE YIELD CONSTRAINT WITH POWER CONSTRAINT RELAXATION

Name	DT	JTS	JTS-DT	(JTS-DT)/DT
AR	74%	90%	16%	22%
DCT	80%	89%	9%	11%
DES	88%	93%	5%	6%
EWF	90%	97%	7%	8%
FF	85%	94%	9%	11%
IIR	83%	90%	7%	8%
Average	83%	92%	9%	11%

- *Comparison against previous variation-aware HLS work.* We also compare our algorithm against previous variation-aware HLS work proposed by Hung et.al [4]. Their module selection is based on a heuristic of using the product of sigma and mean ($\sigma \times \mu$). However, as we have shown in Section III.B, Fig. 2, this heuristic may not be appropriate. In addition, their algorithm only considered the area reduction (using smaller number of resource to meet a specific performance yield) without considering power variations.

VIII. CONCLUSION AND FUTURE WORK

Process variation in deep sub-micron (DSM) VLSI design has become a major challenge for designers. Dealing with delay/power variations during high level synthesis is still in its infancy. Performance/power yield, which is defined as the probability of the synthesized hardware meeting the performance/power constraints, can be used to guide high level synthesis. Our research demonstrates that the yield can be effectively improved by combining both *design-time* variation-aware optimization and *post silicon tuning* techniques (adaptive body biasing (ABB)) during the module selection step in high level synthesis. The experiment results show that significant yield can be achieved compared to traditional worse-case driven module selection technique. To the best of our knowledge, this is the first variability-driven high level synthesis technique that considers post-silicon tuning during design time optimization.

Our future work is to integrate the module selection algorithm into the design flow of a state-of-the-art high level synthesis tool (Catapult C [16]) with the support from Mentor Graphics, and evaluate the effectiveness of our approach using industrial design examples.

REFERENCES

[1] T. Yuan, D. A. Buchanan, C. Wei, D. J. Frank, K. E. Ismail, L. Shih-Hsien, G. A. Sai-Halasz, R. G. Viswanathan, H. J. C. Wann, S. J. Wind, and W. Hon-Sum. Cmos scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, 1997. 0018-9219.

[2] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *Design Automation Conference*, pages 338–342, 2003.

[3] M. Mani, A.K Singh, and M. Orshansky. Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization. In *Proc. of ICCAD*, pages 19–26, 2006.

[4] W.-L. Hung, X. Wu, and Y. Xie. Guarantee performance yield in high level synthesis. In *International Conference on Computer Aids Design*, 2006.

[5] T. Chen and S. Naffziger. Comparison of adaptive body bias (abb) and adaptive supply voltage (asv) for improving delay and leakage under the presence of process variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(5):888–899, 2003. 1063-8210.

[6] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De. Apaptive body bias for reducing imapcts of die-to-die and within-die parameter variaions on microprocessor frequency and leakage. *IEEE Journal of Solid State Circuits*, 37(11):1396–1402, 2002.

[7] N. Azizi and F. N. Najm. Compensation for within-die variations in dynamic logic by using body-bias. In *IEEE-NEWCAS Conference*, pages 167–170, 2005.

[8] S. P. Mohanty and E. Kougianos. Simultaneous power fluctuation and average power minimization during nano-cmos behavioral synthesis. In *Proc. of VLSID*, pages 577–582, 2007.

[9] A. Raghunathan, N. K. Jha, and S. Dey. *High-level power analysis and optimization*. Kluwer Academic Publishers, 1998.

[10] D. Gajski, N. Dutt, and A. Wu. *High-level synthesis: Introduction to chip and system design*. Kluwer Academic Publishers, 1992.

[11] E. Kursun, A. Srivastava, S. G. Memik, and M. Sarrafzadeh. Early evaluation techniques for low power binding. In *International Symposium on Low Power Electronics and Design*, pages 160–165, 2002.

[12] C.-G. Lyuh and T. Kim. High-level synthesis for low power based on network flow method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(3):364–375, 2003. 1063-8210.

[13] X. Tang, H. Zhou, and P. Banerjee. Leakage power optimization with dual-vth library in high-level synthesis. In *Design automation conference*, pages 202–207, 2005.

[14] R. Karri and A. Orailoglu. Time-constrained scheduling during high-level synthesis of fault-secure vlsi digital signal processors. *Reliability, IEEE Transactions on*, 45(3):404–412, 1996. 0018-9529.

[15] R. Mukherjee, S. Ogrenci Memik, and G. Memik. Temperature-aware resource allocation and binding in high-level synthesis. In *Design Automation Conference*, pages 196–201, 2005.

[16] Catapult c synthesis. Technical report, Mentor Graphics Corporation, Products Overview,.

[17] S. Devadas, H. F. Jyu, K. Keutzer, and S. Malik. Statistical timing analysis of combinational circuits. In *International Conference on Computer Design: VLSI in Computers and Processors*, pages 38–43, 1992.

[18] A. Srivastava, D. Sylvester, and D. Blaauw. *Statistical analysis and optimization for VLSI: Timing and power*. Springer, 2005.

[19] S. Sapatnekar. *Timing*. Kluwer Academic Publishers, 2004.

[20] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov. Circuit optimization using statistical static timing analysis. In *Design Automation Conference*, pages 321–324, 2005.

[21] D. Marculescu and S. Garg. System-level process-driven variability analysis for single and multiple voltage-frequency island systems. In *Proc. IEEE/ACM Intl. Conference on Computer-Aided Design (ICCAD)*, Nov. 2006.

[22] S. Garg and D. Marculescu. System-level process variation driven throughput analysis for single and multiple voltage-frequency island designs. *Proc. IEEE Design, Automation and Test in Europe (DATE)*, Apr. 2007.

[23] Jongyoon Jung and Taewhan Kim. Timing Variation-Aware High-Level Synthesis. *Proc. of ICCAD*, November 2007.

[24] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. High-performance cmos variability in the 65-nm regime and beyond. *IBM J. Res. Dev.*, 50(4/5):433–449, 2006.

[25] S.H. Kulkarni, D. Sylvester, and D. Blaauw. A Statistical Framework for Post-Silicon Tuning through Body Bias Clustering. *Proc. of ICCAD*, pages 39–46, Nov. 2006.

[26] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. *IEEE/ACM DAC*, pages 932–937, 2003.

[27] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. *Design Automation Conference (DAC)*, pages 331–336, June 2004.

[28] C. Clark. The greatest of a finite set of random variables. *Operations Research*, pages 145–162, 1961.

[29] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D.Blaauw, and S. Director. Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance. *Design Automation Conference (DAC)*, pages 535–540, 2005.

[30] Anand Raghunathan and Niraj K. Jha. An iterative improvement algorithm for low power data path synthesis. In *Proc. of ICCAD*, pages 597–602, 1995.

[31] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Program.*, 99(2):351–376, 2004.