Abstract—As technology scales down to nanometer regime the process variations have profound effect on circuit characteristics. Meeting timing and power constraints under such process variations in nano-CMOS circuit design is increasingly difficult. This causes a shifting from worst-case based analysis and optimization to statistical or probability based analysis and optimization at every level of circuit abstraction. This paper presents a TED (Taylor Expansion Diagram) based multi–$T_{ox}$ techniques during high-level synthesis (HLS). A variation-aware simultaneous scheduling and resource binding algorithm is proposed which maximizes the power yield under timing yield and performance constraint. For this purpose, a multi–$T_{ox}$ library is characterized under process variation. The delay and power distribution of different functional units are exhaustively studied. The proposed variation-aware algorithm uses those components for generating low power RTL under a given timing yield and performance constraint. The experimental results show significant improvement as high as 95% on leakage power yield under given constraints.

I. INTRODUCTION

As CMOS technology continues to scale down to achieve higher performance and higher level of integration, power consumption and process variation pose new and difficult challenges for integrated circuit designers. The scaling down of technology has resulted in significant deviations from the nominal values of transistor parameters, such as channel length, threshold voltage, and gate-oxide thickness. For example, variation in gate length increases from 35% in a 130 nm technology to almost 60% in a 65 nm technology [1], resulting in the large variation in leakage and performance of the designed circuit.

A lot of work on low-power high-level synthesis (HLS) can be found in the literature. Most of these works [2], [3], [4] have considered dynamic power reduction without considering process variation. But, if the variations are not estimated properly and use existing worst-case analysis, leakage power may exceed the power limit of the design which degrades circuit performance. In [1], authors analyzed the multi–$V_{th}/V_{dd}/T_{ox}$ design space with consideration of process variation at the gate level. A recent work on parametric yield-driven HLS work can be found at [5]. Here, authors present impact of process variations on the multi–$V_{dd}/V_{th}$ techniques at the behavioral level. But, power reduction based on multi–$V_{dd}/V_{th}$ requires extra power supply voltages and is not applicable in performance-critical circuit design. Thus, variation-aware low-power exploration for behavioral synthesis still needs to be investigated further.

The gate-oxide leakage current ($I_{ox}$) in CMOS can be described as:

$$I_{ox} \propto \left(\frac{V_{dd}}{T_{ox}}\right)^2 \exp\left(-\gamma \frac{T_{ox}}{V_{dd}}\right)$$

(1)

where $\gamma$ is an experimentally derived factor. So, $I_{ox}$ is proportional to the square of supply voltage and inversely proportional to the square of $T_{ox}$ (gate-oxide thickness). Reducing supply voltage will increase the delay of the circuit and hence would affect the performance of the design. On the other hand, increase in the gate-oxide thickness leads to increase in propagation delay. So, multiple gate-oxide thickness can serve as a leakage power and delay trade-off. In [6], authors have used dual–$T_{ox}$ based CMOS technology to minimize the leakage current during behavioral synthesis. However, they did not consider delay variations of the functional unit and their RTL generation is not optimal.

The paper presents a variation-aware leakage power optimization work in behavioral synthesis using multi–$T_{ox}$ assignment. In this work, we have used TEDs (Taylor Expansion Diagrams) representation for high-level design description [7], [8] to generate the optimal RTL at the end of synthesis process. This representation is useful for modeling and supporting equivalence verification of designs specified at the behavioral level. TED is a canonical, graph based representation, similar to BDDs (binary decision diagrams) [9] and BMDs (binary moment diagrams) [10]. In contrast to BDDs and BMDs, TED is based on a non-binary decomposition principle, modeled along the Taylor's series expansion. TED is capable of capturing an entire class of structural solutions, rather than a single DFG (data flow graph). By using decomposition, TED can be converted into a structural representation, DFG, optimized for a particular design objective. After obtaining DFG, we do statistical timing and power analysis to determine delay and power distribution through DFG. For this purpose, we explore the impact of process variation on delay and leakage power. A variation-aware resource library is constructed where all the library units
are characterized based on their delay and power distribution at different $T_{ox}$. A variation-aware simultaneous scheduling and resource binding algorithm is presented which takes time constraint as a performance (or delay) trade-off factor and offers user to maximum leakage power yield. The algorithm schedules nodes of DFG at the appropriate control steps and simultaneously binds them to the best available resources while considering resource constraint so as to achieve the desire performance with maximum leakage power yield. The contributions of the paper can be summarized as,

- To best of our knowledge so far, this is the first work to use TED techniques during behavioral synthesis in presence of both delay and leakage power variation.
- The HLS flow for variation-aware leakage power optimization in multi$-T_{ox}$ is proposed.
- Consideration of both resource and time constraints to provide user an optimal RTL by taking account of process variations.

II. POWER, LEAKAGE, DELAY, AND YIELD TRADE-OFFS AT RTL

In this section, we mainly discuss some preliminaries on variation-aware high-level synthesis (HLS), and present the motivation of our work.

A. Timing and Power Yield in HLS

HLS is a process of translating a behavior description into a register level structural description. Scheduling and resource binding are key steps during the synthesis process. The scheduler divides the set of arithmetic and logical operation in the DFG into groups so that the operations in the same group can be executed concurrently, while taking into consideration possible trade-offs between total execution cost and hardware cost. The binding process selects resources from the library, which involves trade-offs according to different features like delay, area, power, and leakage. The resource library contains different functional units with different characteristics such as delay, leakage, etc. Traditional HLS algorithms consider worst-case latency of each functional unit during scheduling and binding. However, as the magnitude of process variation grows rapidly, worse-case based analysis and optimization are no longer acceptable since they introduce too much pessimism in the design. This in turn creates problem for designers to meet the requirement. Instead, statistical description and analysis of functional units are introduced to tackle the timing problem [11], [12], [13], [14], [15], [16], [18].

In presence of process variation, the delay of the functional unit is no longer a fixed value, but spreads into wide distributions. In a statistical timing view, the distribution can be described by a probability density function (PDF). Timing yield is defined as the probability that a functional unit can finish execution in a given time period. Alternatively, it is the cumulative probability under a given $T_{clk}$ in PDF. An concept of timing yield is shown in Fig. 1

Given the clock time $T_{clk}$, the overall timing yield of the entire DFG is the probability that the entire design can finish execution within $T_{clk}$, and can be defined as

$$Yields = P(t_1 \leq T_{clk}, t_2 \leq T_{clk}, \ldots, t_n \leq T_{clk})$$

where $P()$ is the probability function, $t_1$, $t_2$, ..., $t_n$ are the execution time for the control steps 1, 2, ..., $n$, respectively.

In variation-aware HLS, a metric called parametric yield is introduced in [5]. The parametric yield is defined as the probability of the synthesized hardware meeting a specified constraint $Yield = P(Y \leq Y_{max})$, where $Y$ can be delay and power.

Fig. 2 shows an example to compare yield-driven approach and worse-case deterministic approach. Four functional units $F_1$, $F_2$, $F_3$, and $F_4$ have the same functional description. However, $T_{ox}$ (gate-oxide thickness) of these functional units increases from $F_1$ to $F_4$. The leakage power and delay distribution of these units are shown in Fig. 2. The power limit $P_L$ and clock cycle time $T_{clk}$ are also shown in Fig. 2. The $T_{ox}$ of $F_1$, $F_2$, $F_3$, and $F_4$ follows $T_{ox}(F_1) < T_{ox}(F_2) < T_{ox}(F_3) < T_{ox}(F_4)$. So, mean leakage power follows $\mu(F_4) < \mu(F_3) < \mu(F_2) < \mu(F_1)$ and delay follows up as $\mu(F_1) < \mu(F_2) < \mu(F_3) < \mu(F_4)$. In worse-case deterministic
approach F4 will be chosen under leakage power constraint as it has lowest leakage power consumption. But, from statistical point of view F4 has low timing yield and may cause timing violation. Similarly, from the performance constraint point of view F1 will be chosen as it satisfies timing constraint. But, F1 has larger leakage power and may lead to higher power dissipation. However, if we consider both power and performance constraint simultaneously, F2 and F3 can be chosen. Selection of F3 results in slightly loss in timing yield but satisfies power yield, while F2 results in slightly loss in power yield but satisfy timing yield. So, selection of F2 and F3 introduces a concept of tradeoff in between timing and power yield. Thus, a yield-driven statistical approach is needed which selects the functional units so that one parameter yield can be maximized under other parametric yield constraint.

B. Library setup for yield-driven multi – $T_{ox}$ optimization

Leakage power is inversely proportional to the gate-oxide thickness ($T_{ox}$). The reduction in $T_{ox}$ results increase in leakage power and decrease in delay. In the present work, we have created libraries with different $T_{ox}$ components. For this purpose, we first characterized a library of 16 – bit components, such as adders, subtractors, multipliers, comparators, multiplexers, and registers following the structural description from [19]. We performed our library simulation using different gate-oxide thicknesses. The Berkeley Predictive Technology Model (BPTM) of 45nm technology node is used in this work, with base values of $T_{ox} = 1.4nm$, $V_{dd} = 0.7V$ and $V_{th} = 0.22V$. The nominal power supply is $V_{dd} = 0.7V$. The effect of varying oxide thickness was incorporated by varying the parameter $t_{ox}$ in the SPICE model deck directly. It may be noted that the length of the device is proportionately changed to maintain a constant ($L/T_{ox}$) ratio in order to minimize the impact of higher oxide thickness on device performance and to maintain the per width gate capacitance constant as per fabrication requirements [17]. The PMOS transistors are sized appropriately to ensure proper functionality of the building blocks. We have exhaustively evaluated the process variation effects through detailed 10000 Monte Carlo simulations to capture the effects. The primary goal of this analysis is to assess the extent of leakage $I_{ox}$ and power variation as a result of process variations in gate oxide thickness $T_{ox}$. The distribution of these parameters is assumed to be Gaussian with the variance of 10%. Table I shows the statistical variation of the delay corresponding to oxide thickness 1.4nm and 1.7nm respectively.

Table I indicates the delay values of the functional units under different performance yields. In order to obtain these values, we first generate the delay distribution for each functional unit. The delay for certain timing yield can be calculated by finding the area under the curve. For example, Fig. 3 shows the PDF for the adder of Table I. Under 100% yield, the delay of the adder is 11.68 $ns$ ($T_{ox}$ = 1.4 $nm$) which corresponds to point A in Fig. 3. But, if we sacrifice 10% yield, the delay becomes 10.94 $ns$ which corresponds to point B. Similarly, points C and D represent the delay value of 11.09 $ns$ and 10.98 $ns$ correspond to 97% and 94% of timing yield.

![PDF](image)

**Fig. 3. The delay values of an adder under different timing yield**

Similar to delay variation, we have generated the leakage power under different power yield for each functional unit which is not shown here due to space limitation. Once characterized, the next task is to create the $DFG$ from the behavioral description of the given circuit. In order to create optimized $DFG$, in the present work we have used TED based approaches which is described in the next section.

### III. Canonical TED for Efficient High-Level Representation

Taylor Expansion diagram [8] is a canonical, word-level data structure that offers an efficient way to represent computation in a compact, factored form. An Algebraic, multi-variable expression $f(x,y,...)$, can be represented using Taylor series expansion, w.r.t. variable $x$ as follows:

$$f(x,y,...) = f(x=0) + xf'(x=0) + 1/2x^2f''(x=0) + ...$$

(4)

Where $f'(x)$, $f''(x)$, etc, are the successive derivatives of $f$ w.r.t. $x$. The terms of the decomposition are then decomposed with respect to the remaining variables ($y,...,etc$), one variable at a time. A directed acyclic graph is used to store the resulting decomposition whose nodes represent the terms of the expansion. The detailed explanation of TED can be found in [7], [8].

**A. TED – based RTL low-leakage optimization: A Finite Impulse Filter (FIR) Case Study**

Since FIR (Finite-impulse response) filters are critical to most DSP application, an energy-aware filter design helps significantly in reducing the total power dissipation. The

<table>
<thead>
<tr>
<th>Functional unit</th>
<th>$I_{ox}$ $(\mu A)$</th>
<th>$T_{pd}$ $(ns)$</th>
<th>Yield 100%</th>
<th>Yield 90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adder</td>
<td>2.155</td>
<td>11.68</td>
<td>14.92</td>
<td>13.12</td>
</tr>
<tr>
<td>Subtractor</td>
<td>11.99</td>
<td>11.46</td>
<td>13.15</td>
<td>13.15</td>
</tr>
<tr>
<td>Multiplier</td>
<td>53.81</td>
<td>15.55</td>
<td>16.45</td>
<td>16.45</td>
</tr>
<tr>
<td>Comparator</td>
<td>3.30</td>
<td>0.2304</td>
<td>0.2396</td>
<td>0.2396</td>
</tr>
<tr>
<td>Register</td>
<td>3.465</td>
<td>0.7934</td>
<td>0.7973</td>
<td>0.7973</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>3.181</td>
<td>0.3763</td>
<td>0.3997</td>
<td>0.383</td>
</tr>
</tbody>
</table>

**TABLE I**

<table>
<thead>
<tr>
<th>Library with different gate-oxide thickness (nm)</th>
<th>$I_{ox}$ $(\mu A)$</th>
<th>$T_{pd}$ $(ns)$</th>
<th>Yield 100%</th>
<th>Yield 90%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ox} = 1.4nm$</td>
<td>2.155</td>
<td>11.68</td>
<td>14.92</td>
<td>13.12</td>
</tr>
<tr>
<td>$T_{ox} = 1.7nm$</td>
<td>53.81</td>
<td>15.55</td>
<td>16.45</td>
<td>16.45</td>
</tr>
</tbody>
</table>

**TABLE II**
polynomial corresponding to a 4–tap FIR filter can be written as,

\[ Y_n = a_0 X_n + a_1 X_{n-1} + a_2 X_{n-2} + a_3 X_{n-3} \]  \hspace{1cm} (5)

TED corresponding to equation 5 is shown in Fig. 4. Given an optimized TED, the next task is to convert it to DFG, shown in Fig. 5. An STA on DFG is performed to generate the necessary timing information. Specifically, we need to calculate arrival time \( T_a \), required time \( T_r \), and slack \( T_s = T_r - T_a \), for each node.

**Definition 1:** Arrival time \( T_a \) of a DFG node \( n \) is recursively defined as a sum of delay of node \( n \) and the maximum arrival time of its inputs:

\[ T_a(n) = \text{Delay}(n) + \max(T_a(n_i) | n_i \in \text{Input}(n)) \]  \hspace{1cm} (6)

where Delay\((n)\) denotes the delay of the operation associated with node \( n \), and Input\((n)\) is the set of input nodes to the node \( n \).

**Definition 2:** Required time \( T_r \) of a node \( n \) is recursively defined as a difference between the minimum required time of its outputs and delay of node \( n \):

\[ T_r(n) = \min(T_r(n_o) | n_o \in \text{Output}(n)) - \text{Delay}(n) \]  \hspace{1cm} (7)

Here Output\((n)\) is the set of output DFG nodes of node \( n \).

**Definition 3:** Slack time \( T_s \) of a DFG node \( n \) is defined as a difference between its required time \( T_r \) and the arrival time \( T_a \).

\[ T_s(n) = T_r(n) - T_a(n) \]  \hspace{1cm} (8)

In Fig. 5, the arrival time \( T_a \), the required time \( T_r \), and the slack \( T_s \) of each node are denoted in the form of \( [T_a/T_r/T_s] \). Here, we assume delay of each functional unit is 1 for simplicity. Based on the definition of slack, a critical node and critical path in DFG can be identified as follows.

**Definition 4:** A critical node in a DFG is a node which has a slack equal to 0. A critical path is a path which contains critical nodes only.

In Fig. 5, critical path 1 consists of 4 nodes (\( M_1, A_1, A_2, A_3 \)) and critical path 2 consists of 4 nodes (\( M_2, A_1, A_2, A_3 \)). In the next subsection, we present the generalized algorithm for variation-aware simultaneous scheduling-binding for general circuits.

**B. An Algorithm for variation-aware Nano–CMOS RTL leakage optimization**

In this section, we present a simultaneous scheduling and binding algorithm under resource constraint. The inputs to the algorithm are an unscheduled DFG, libraries with different resources made of transistors of different gate-oxide thickness, a delay trade-off factor \( T_d \), and performance yield \( Y_d \). The \( T_d \) is a user-defined quantity which specifies the maximum allowed critical path delay of the target circuit. The present algorithm schedules the DFG in such a way that critical path delay is either less than or equal to \( T_d \) while improves the power yield under performance yield constraint \( Y_d \).

The proposed algorithm performs an initial STA on the DFG to identify critical and non-critical nodes by calculating \( T_a, T_r, \) and \( T_s \) of each node. During this step, it uses delay value of 1 for each node. Once identified, it assigns \( T_{oxL} \) (low gate-oxide component) to critical nodes and \( T_{oxH} \) (high gate-oxide components) to non-critical nodes. After initial scheduling and binding, it calculates critical path delay and power yield. Now the algorithm searches iteratively on the DFG to reduce the leakage power or improves the power yield under \( Y_d \) and \( T_d \) constraints. Or in other words, it replaces nodes of \( T_{oxL} \) with \( T_{oxH} \) so that power yield can be maximized by satisfying \( Y_d \) and \( T_d \). The pseudo code of the algorithm is presented in Algorithm 1.

Consider the FIR filter of Fig. 4 under the assumption that unlimited number of \( T_{oxL} \) and \( T_{oxH} \) components. The \( T_d \) and \( Y_d \) are assumed to 10 ns and 90% respectively. The delay of the library units under 100% and 90% timing yield is shown in Table II for both \( T_{oxL} \) and \( T_{oxH} \) components.

**TABLE II**

<table>
<thead>
<tr>
<th>Functional Unit</th>
<th>( \text{Delay}(n) ) for ( T_{oxL} )</th>
<th>( \text{Delay}(n) ) for ( T_{oxH} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adder</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Multiplier</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Fig. 6 shows the DFG of equation 6 after initial scheduling and binding where critical nodes are bound to \( T_{oxL} \) and non-critical nodes to \( T_{oxH} \). During this phase, the algorithm uses delay values of the functional units corresponding to 100% timing yield (or worse-case analysis). Once scheduled, the
Algorithm 1 NanoCMOS RTL Optimization for Yield, Power, and Time Tradeoff

1: Apply STA to DFG under resource constraint
2: Assume each node is assign to a delay of 1
3: Identified critical and non-critical nodes
4: for all critical nodes \( n_i \) do
5: if \( FU_j(k, T_{oxL}) \) is available for control step \( C[n_i] \) then
6: Assign \( FU_j(k, T_{oxL}) \) to node \( n_i \)
7: else
8: Assign \( FU_j(k, T_{oxH}) \) to node \( n_i \)
9: end if
10: end for
11: for all non-critical nodes \( n_i \) from root of the DFG do
12: for all possible control steps (slack) of \( n_i \) do
13: if \( FU_j(k, T_{oxH}) \) is available for control step \( C[n_i] \) then
14: schedule \( n_i \) in control step \( C[n_i] \)
15: Assign \( FU_j(k, T_{oxH}) \) to node \( n_i \)
16: Update \( T_s \) for all the nodes connected to \( n_i \)
17: end if
18: end for
19: if \( n_i \) is not scheduled then
20: for all possible control steps (slack) of \( n_i \) do
21: if \( FU_j(k, T_{oxH}) \) is available for control step \( C[n_i] \) then
22: schedule \( n_i \) in control step \( C[n_i] \)
23: Assign \( FU_j(k, T_{oxH}) \) to node \( n_i \)
24: Update \( T_s \) for all the nodes connected to \( n_i \)
25: end if
26: end for
27: end if
28: end for
29: Calculate \( T_a \), \( T_r \), and \( T_s \) for all nodes
30: Calculate critical path delay \( T_{cp} \) and power yield \( Y_p \) of the DFG
31: Sort all critical nodes according to ascending order of leakage current
32: Calculate timing yield \( Y_t \) of the DFG
33: for all critical nodes \( n_i \) do
34: if \( (T_a + T_{cp}) \) and \( (Y_t + Y_p) \) then
35: if \( FU_j(k, T_{oxH}) \) is available for control step \( C[n_i] \) then
36: Assign \( FU_j(k, T_{oxH}) \) to node \( n_i \)
37: Calculate \( Y_t \) and modified power yield \( Y_{pt} \) of DFG
38: if \( ((Y_{pt} - Y_p) \) greater than 0) and \( (Y_t \) greater than \( Y_d) \) then
39: calculate \( T_{cp} \)
40: if \( T_{cp} \) less than or equal to \( T_d \) then
41: \( Y_p = Y_{pt} \)
42: continue to next critical node
43: end if
44: end if
45: Assign \( FU_j(k, T_{oxH}) \) to node \( n_i \)
46: end if
47: end if
48: end for

algorithm searches iteratively to bind \( T_{oxL} \) node with \( T_{oxH} \) components under \( Y_d \) and \( T_d \) constraints. It is clear from Fig. 6 that \( M1 \) can be replaced with \( T_{oxH} \) as under 90% yield delay of the multiplier is 4ns (see Table II). After replacement, timing yield of the DFG will be 90% which is equal to \( Y_d \). Similarly, instead of \( M1 \) one can also replace \( A3 \) with \( T_{oxH} \) components. But, replacement of multiplier saves much more leakage power than an adder. The final scheduled DFG is shown in Fig. 7.

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results of our variation-aware leakage power yield improvement framework for HLS. We implement our variation-aware synthesis algorithm in C and perform the experiment on a set of HLS benchmarks [6]. The results show that our method can effectively improve the overall leakage power yield or minimize the leakage power dissipation under process variation.

Table III shows the comparison of variation-aware resource binding algorithm against traditional deterministic (worse-case) approach. Gate-oxide leakages current \( (I_{ox}) \) have been calculated under different performance yield \( (Y_d) \) and \( T_d \) constraints including worse-case delay based approach (i.e., when \( Y_d = 100\% \)). The results indicate significant reduction in \( I_{ox} \) when we scarify 10% of performance yield.

Fig. 8 shows the leakage power yield improvement against worst-case delay based approach under timing yield constraints 95% and 90% for different benchmark circuits when \( T_d = 1.2ns \). Results indicate that the power yield improvement
depends on how much timing yield loss is affordable for the design.

Fig. 8. Leakage power yield improvement against worst-case approach.

Since there is no behavioural synthesis research dealing with gate-oxide leakage reduction for different performance yield, direct comparison is not possible. However, in view of power yield improvement we provide a broader comparative perspective in Table IV with [5]. $\Delta p$ indicates the power yield improvement against deterministic worst-case approach under different performance yield. From Table IV, it is clear that multi $- \Delta p$ is an attractive approach for improving performance yield at the same time reducing gate leakage current for nano-CMOS datapath circuits.

### V. Conclusions

In this paper we presented an effective way to the problem of variation-aware HLS. Here, we develop the timing and leakage power constraints based scheduling and resource binding algorithm for HLS. We have used TED based approaches to generate the optimize DFG. The proposed algorithm maximizes the leakage power yield of the design circuit for a given performance constraint. The experimental results on several benchmark circuits show that performance yield can be maintained with increasing leakage power yield.

### VI. Acknowledgments

The work was partially funded by the Engineering and Physical Science Research Council (EPSRC), United Kingdom, under Grant No: EP/G032904/1 and EP/F030991/1. Author S. P. Mohanty acknowledges NSF award CNS $- 0854182$ for partial support for this research.

### References