Statistical High-Level Synthesis under Process Variability

Yuan Xie and Yibo Chen
Pennsylvania State University

Editor's note:
CMOS process variability is a major challenge in deep-submicron SoC designs. The variations in transistor parameters are complicating both timing and power consumption prediction. This article surveys recent progress in the statistical high-level synthesis area.

—Philippe Coussy, Université de Bretagne-Sud

It is a truism that technology scaling improves chip performance and increases integration density. The latest Intel Xeon Nehalem-EX processor, for example, is built on the most advanced 45-nm technology, which has resulted in a single chip's being populated with 2.3 billion transistors. The semiconductor industry expects to use 32-nm technology in volume production by the end of 2009, which will mean even higher transistor density on a chip. Such dense integration onto a single chip with nanoscale transistors has posed two major challenges for chip designers. The first challenge is the increasing gap between productivity and design complexity. Consequently, we have witnessed a recent trend toward moving design abstraction to a higher level, with an emphasis on electronic system-level (ESL) design methodologies. These methodologies will likely be enabled by high-level synthesis, which is the process of translating a behavioral description into an RTL description. For example, many companies have used the Mentor Graphics HLS tool, Catapult C, in the design flow. Other popular HLS tools include Bluespec's suite of tools and Cadence Design Systems' C-to-Silicon.

The second challenge confronting designers is process variability, which causes considerable fluctuations in performance and power. The complexities in fabricating transistors with a very small feature size have caused significant variations in transistor parameters (such as transistor channel length, gate-oxide thickness, and threshold voltage) across identically designed neighboring transistors (*within-die variation*) and across different identically designed chips (*interdie variation*). These manufacturing uncertainties can cause substantial performance and power variations in chip design. For example, Intel has reported a 30% variation in chip frequency and a 20× variation in chip leakage in 1,000 sample chips fabricated in 180-nm technology.[1] In 45-nm technology, the relative process variations are reportedly even worse.[1] Consequently, dealing with variability has become one of the major design focuses for nanoscale VLSI design.

Traditionally, performance and power variations have been addressed through a combination of speed and power binning and design margining. Speed and power binning tests all fabricated chips—those 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 meeting the design requirement. These solutions are rapidly becoming inadequate, however, as variability increases along with technology scaling, and they might not be viable when the variability that designers encounter in the new process technologies exceeds an acceptable norm. Moreover, cost sensitivity makes designing for the worst-case manufactured hardware unacceptable. Consequently, a shift in the design paradigm, from today's deterministic design to statistical or probabilistic design, is critical for deep-submicron design. Industry and academia have already realized the need for such a shift in the design paradigm. Consequently, we have

0740-7475/09/$26.00 © 2009 IEEE Copublished by the IEEE CS and the IEEE CASS

encountered considerable research on statistical variation-aware design methodologies.[2]

The purpose of this article is to survey recent progress in statistical HLS, discuss possible future directions for the technique, and serve as a catalyst to accelerate future research in this novel direction—especially as it pertains to resolving problems caused by process variability. The majority of the existing analysis and optimization techniques related to process variations occur at the lower (device or logic gate) level. In the domain of HLS, process-variation-aware research is still in its infancy. It is important to raise 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 consider lower-level process variations, avoiding late surprises and possibly expensive design iterations. Such statistical HLS research motivated us to pursue a novel research direction: investigating the impact of process variations at early design stages, by moving the ESL design methodologies from deterministic to probabilistic design. Statistical HLS will provide a complementary perspective to the existing research on statistical analysis and optimization at lower device design or logic design levels, and bring the awareness of process variation to the ESL design paradigm.

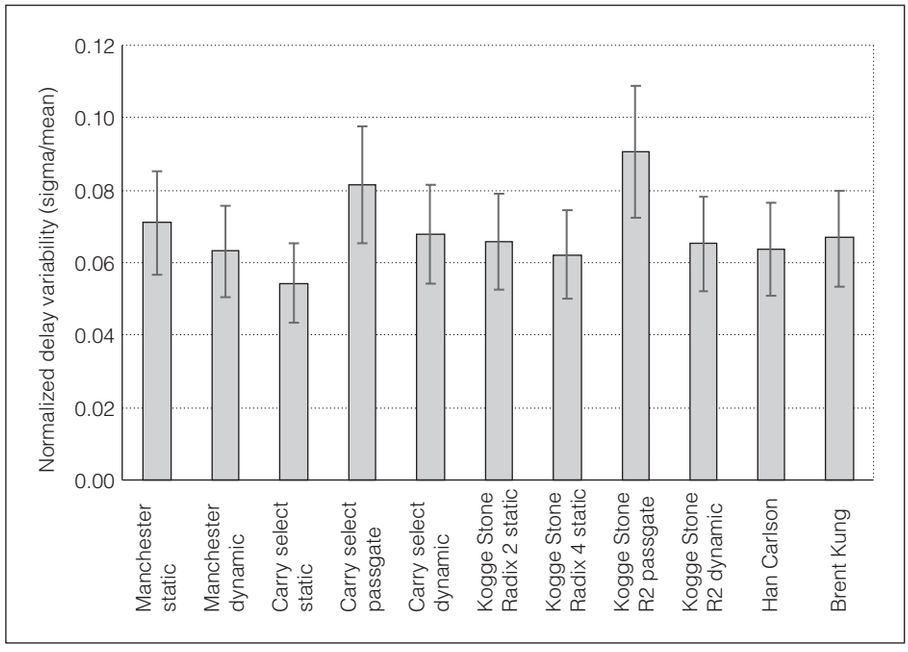

Figure 1. Delay variation (normalized ratio of sigma to mean) for 16-bit adders in IBM's Cu-08 (90 nm) ASIC technology (Courtesy of Kerry Bernstein, IBM).

Process variations: Influence on HLS

In HLS, the design specification is usually written as a behavioral description that is translated into an internal representation such as parse trees or control dataflow graphs (CDFGs), which are then mapped to the functional units selected from the hardware resource library to optimize design goals (such as performance, area, and power). The HLS process usually consists of module selection, scheduling, resource binding, and clock selection.[3]

Traditionally, worst-case delay and power parameters for each resource have facilitated design space exploration in HLS. However, greater variability in new process technologies is making these parameters unsuitable. For example, Figure 1 shows the delay variations (depicted as normalized sigma and mean) for 11 different types of 16-bit adders that span a range of circuit architecture and logic evaluation styles. Under the influence of process variation, the existing deterministic worst-case design methodologies in HLS can cause designers to overestimate the resources needed to meet the performance goal, resulting in an unexpected performance discrepancy, a pessimistic performance and power estimation, or the use of excess resources to guarantee design constraints.

Figure 2 shows, as a result of resource sharing, a multiplexer and an adder that have been connected in a cascade. We'll assume that the delay of the adder (D_{add}) and the multiplexer (D_{mux}) conform to independent Gaussian distributions $N(\mu, \sigma)$, where μ and σ are the mean and standard variance of the delay distribution. Values for these parameters are as shown in the figure.

In conventional worst-case analysis, the worst-case execution time (WCET) is calculated as $(\mu + 3\sigma)$. So the execution time for the data path is $(\mu D_{mux} + 3\sigma D_{mux} + \mu D_{add} + 3\sigma D_{add})$, which is 91 ps. However, based on the statistical information,

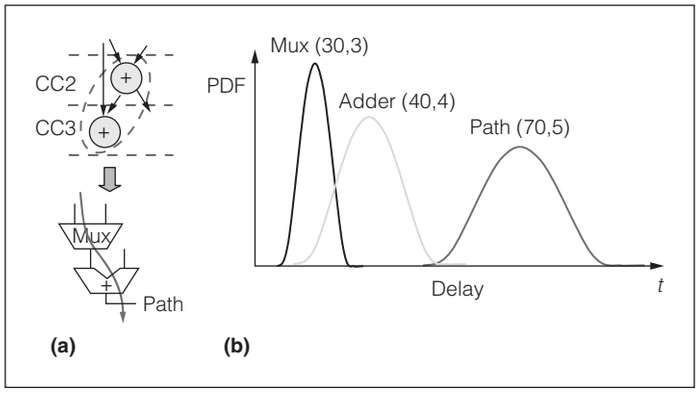

Figure 2. Comparison of approaches based on worst-case execution time (WCET) and statistical analysis. The example dataflow graph and the underlying hardware mapping (a); the delay distributions of the underlying hardware (b). (PDF: probability density function.)

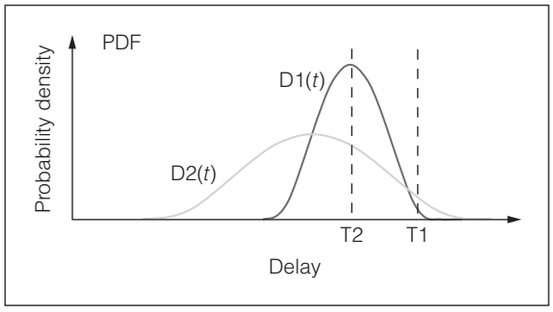

Figure 3. An example illustrates the effectiveness of the performance yield metric. D_1 and D_2 are path delay distributions of two different synthesized results with the same functionality; T_1 and T_2 are clock cycle times.

the path's delay follows a new Gaussian distribution:

$$N(\mu D_{\text{add}} + \mu D_{\text{mux}}, \sqrt{\sigma^2 D_{\text{add}} + \sigma^2 D_{\text{mux}}})$$

and the 3σ delay of the path is 85 ps. Compared with 91 ns obtained using WCET analysis, the statistical approach achieves a tighter estimation of the circuit's performance.

The previous example shows the benefit of calculating the worst-case delay using a statistical approach. However, in the statistical design era, a better metric for evaluating circuit performance or power is needed instead of the deterministic worst-case delay or power values.

To bring the process-variation awareness to the HLS flow, researchers have proposed the concept of *parametric yield*,[4] for which the parameter can be either performance or power. The *performance yield* is defined as the probability of the synthesis results meeting the clock cycle time constraints under the latency 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.

Figure 3 demonstrates the effectiveness of the performance yield metric in a simple example. A delay distribution is usually expressed in the form of probability density function (PDF), as shown in the figure. The performance yield of a design for a given clock cycle time T is then calculated as the area of the region between the PDF curve of the design and the x axis, cut off at the clock cycle time T. In this example, assume we have two synthesized results with the critical path delay distributions of $D_1(t)$ and $D_2(t)$. When the clock cycle time is set to T_1, the synthesis result with $D_1(t)$ is better than that with $D_2(t)$ in terms of performance yield. However, when the clock cycle time is set to T_2, the synthesis result with $D_2(t)$ is better than that with $D_1(t)$. If we use the worst-case delay to evaluate the results, we would always choose the synthesis result attained with $D_1(t)$.

The parametric yield of the HLS-resultant hardware depends on all the HLS steps: scheduling, module selection, resource sharing, and clock selection. For example, in Figure 4, the same dataflow graph (DFG) can be either scheduled into four clock cycles (CC1 through CC4 in Figure 4a) with a clock cycle time of T_S, or scheduled into two clock cycles (CC2 through CC2 in Figure 4b) with a longer clock cycle time, T_L.

The computation time is $4 \times T_S$ and $2 \times T_L$, respectively. In Figure 4a, the synthesized underlying architecture would be one multiplier and one adder; in Figure 4b, the synthesized underlying architecture would be two adders and one multiplier, with the adders and multiplier connected in series (for illustration purposes, the possible multiplexers and registers are omitted in this example). Because the delay distributions of the adder and the multiplier are independent of each other, the performance yield of Figure 4a is calculated as Equation 1:

$$Y_\alpha = \int_0^{T_S} D_{\text{adder}}(t)dt \times \int_0^{T_S} D_{\text{mult}}(t)dt \qquad (1)$$

where D_{adder} and D_{mult} are the probability density function (PDF) for the adder and the multiplier, respectively. What does the subscript α mean? Is it itself a variable (numbers could be substituted for it)? The PDF for the functional units are affected by the module selection step in HLS. In Figure 4b, because both add operations are scheduled in the same clock cycle, resource sharing with one single adder is not possible, so two adders are needed. In addition, both `mult` and `adder1` are connected to `adder2`, which will not start execution until the outputs of both `mult` and `adder1` are available. The total delay in this clock cycle is the `max` delay of `mult` and `adder1`, plus the delay of `adder2`. Therefore, a `max` operation is applied to the delay distributions of `mult` and `adder1`, resulting in a new *maximum-delay* distribution of these two operations. Consequently, the overall performance yield is calculated as Equation 2:

$$Y_b = \int_0^{T_L} (D_{adder2}(t) \times \int_0^{T_L - t} D_{max}(s)ds)dt \qquad (2)$$

where D_{max} is the maximum distribution of `mult` and `adder1`. The equation shows that, when `adder2` needs time t (which varies from 0 to T_L) to finish execution, `adder1` and `mult` could have maximum execution time of $T_L - t$, in order not to violate the timing of that cycle.

The examples in Figures 3 and 4 simply illustrate that the parametric yield of the HLS-resultant hardware depends on all HLS steps (scheduling, module selection, resource sharing, and clock selection), which are usually tightly coupled with one another during HLS, and influence the design's final parametric yield.

Key issues in statistical HLS

Statistical HLS implies a shift to statistical design methodologies for all the stages of high-level synthesis. We have identified the key issues involved in the shift and describe them here.

Characterization and statistical analysis

To facilitate design space exploration while considering process variations, the resource library of functional units for HLS must be characterized for delay and power variations. Under the influence of process variations, each component's delay and power are no longer fixed values but are represented by a PDF. Consequently, the characterization of functional units with delay and power variations requires statistical analysis methodologies.

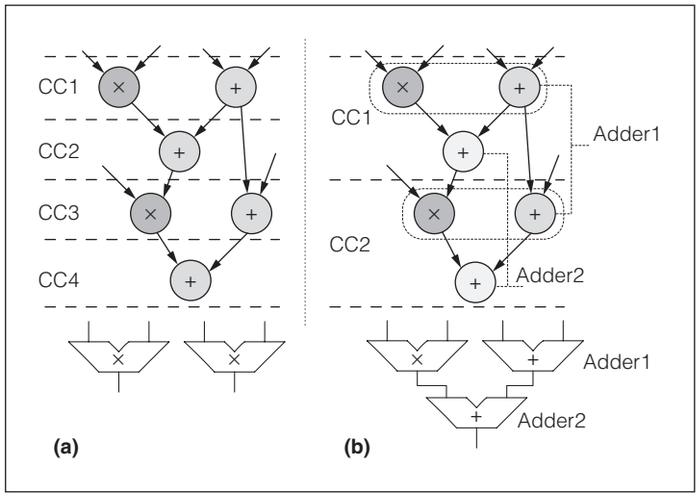

Figure 4. An example of different scheduling, resource sharing, and clock selection affecting performance yield. The dataflow graph is scheduled into four clock cycles (a) and two clock cycles (b).

Delay characterization. Gate-level statistical-timing-analysis tools, such as the Synopsys PrimeTime VX or the IBM Einstimer/Einstat, can be used to characterize delay variations. These variation-aware timing tools increase the accuracy of timing analysis by considering the statistical distribution of device parameters (such as channel length and gate-oxide thickness). For example, by using PrimeTime VX we can characterize the delay PDF of functional units with the following steps.

1. All the gates in a standard cell library (such as NAND or NOR gates) for a specific technology node (such as 45 nm) are characterized using the Synopsys gate-level characterization tool NCX.
2. The functional units in the HLS resource library are then synthesized to a gate-level netlist with the standard cell library.
3. Statistical timing analysis for the functional units is performed using PrimeTime VX, and the parameters of delay distributions are reported.

Power characterization. Statistical power characterization for functional units in the resource library can be accomplished by using Monte Carlo analysis in Spice. The power consumption of functional

units consists of dynamic and leakage power. Although dynamic power is relatively immune to process variation, leakage power is greatly affected, and it predominates as the technology continues to scale down. Consequently, leakage power for each gate in the standard cell library is characterized using Monte Carlo analysis. Similar to delay characterization, after each component in the HLS resource library is mapped to the standard cell library as a gate-level netlist, we can estimate the power PDF via the gate-level power characterization.

Statistical timing and power analysis

Similar to the way that Srivastava et al. modeled gate-level delay in a statistical manner,[2] we can use a first-order canonical model to model the delay for a functional unit in the resource library, rather than using a PDF. In this model, we express the delay of a gate as

$$D_m = d_0 + \sum_{i=1}^{n} d_i X_i + d_{n+1} X_m \quad (3)$$

where d_0 is the nominal delay of a component in the resource library. The independent, normally distributed random variables X_i and X_m model the variations in the process parameters: X_i is the correlated component of these variation parameters, such as channel length, gate-oxide thickness, metal-line width, and so on; and X_m is the purely random component.

In the statistical timing analysis, the `max` and `sum` operations propagate the delay distribution in the synthesized results of HLS.[2] Although the `sum` distribution can be computed by accumulating the corresponding normal distributions, the `max` distribution can be computed using tightness probability and moment matching. We maintain the results of these two operations in canonical form. Consequently, the circuit's delay is also expressed in canonical form. Thus, the functional units' delay in HLS can be expressed in canonical form as shown in Equation 3. For instance, in Figure 2b, the `max` operation is performed over the delay values of the first adder and multiplier to obtain the delay distribution before the input to the second adder; the `sum` operation is performed over the first-stage delay and the second adder's delay to obtain the path delay for the entire clock step. The results of the `max` and `sum` operations are also expressed in linear canonical form.[4]

Similar to the way statistical leakage power was calculated in the gate-level statistical power analysis by Srivastava et al.,[2] we can express the statistical leakage power of a functional unit as

$$P_m = \exp(a_0 + \sum_{i=1}^{n} a_i X_i + a_{n+1} X_m) \quad (4)$$

where exp is the exponentiation function and $\exp(a_0)$ is the nominal leakage power of a functional unit. Similarly, X_i and X_m are the independent normally distributed random variables for modeling the variations in process parameters. We compute the circuit's total power as the summation of the power of all components in the circuit. The summation result is also expressed in the same form as Equation 4.[2] We compute the total power of the synthesized results in HLS by iteratively adding the leakage power of the function units.

Variation-aware HLS

With the characterized variation-aware resource library and the statistical analysis methods, designers can use the yield-driven HLS algorithms to perform design space exploration statistically, and search for solutions to improve performance yield, power yield, or both.

One of the early attempts in variation-aware HLS, made by Hung et al., introduced the performance yield concept.[5] Hung et al. based the HLS framework on a simulated annealing engine, and based the statistical timing analysis on discrete delay distribution for each component in the resource library. They proposed a variation-aware clock cycle time selection algorithm to improve the use of clock slacks and reduce the sensitivity to timing variations. They also integrated the performance yield into the cost function to guide the synthesis. However, other steps they took (such as scheduling, and resource binding and sharing) remained conventional. Their work demonstrated that integrating performance yield into the design space exploration can satisfy the yield requirement, and they achieved a resource reduction of 14% compared to conventional HLS approaches.

Jung and Kim used a similar statistical discrete-delay-distribution analysis method, and proposed a heuristic algorithm for variation-aware scheduling and resource binding.[6] Their heuristic focuses on improving performance yield under latency constraints by iteratively searching the DFG for *yield-equivalent* operation

patterns and binding them to the same combination of resource units. Within these yield-equivalent patterns, the work implicitly uses a *time-borrowing* technique via operation chaining, which improves performance yield by sharing timing slacks between faster and slower operations. Moreover, enhanced resource sharing reduces the module usage and consequently improves the performance yield, given that the overall performance yield is approximately the product of the yields of all module instances used in the design.

We extended the statistical-analysis-based HLS framework of Hung et al. to modify all the HLS steps such that they are variation aware,[4,5] and we integrated statistical power analysis. This framework optimizes both performance yield and power yield during the HLS design iteration. The framework takes a DFG, constraints (latency, resource, clock cycle time, and power), and a resource library as inputs, and generates a synthesized DFG that is power optimized while satisfying the performance constraints. Because the HLS subtasks are strongly interdependent and affect the parametric yield (as shown in Figure 2), this work brings performance and power variation awareness into the HLS subtasks by simultaneously performing yield-driven module selection, resource sharing, and scheduling. Our experiments show that the yield-driven HLS framework can achieve on average a 31% power yield improvement with only 1% of performance yield loss, compared to the traditional worst-case-analysis-based approach.

Both heuristic algorithms (such as statistical-analysis-based algorithms) and integer linear programming (ILP)-based algorithms can be used to solve the scheduling and resource binding problem in HLS.[3] Furthermore, we have extended the conventional ILP-based HLS framework and have proposed applying a logarithm operation on the yield calculation equation so that it can be handled by any LP solvers.[7] Although the number of variables and inequalities in an ILP formulation grows exponentially as the problem size scales up, such ILP-based variation-aware HLS algorithms can be used to find an optimal solution as a reference for evaluating the performance of different heuristic algorithms, which are more practical than ILP-based algorithms in real designs.

Adding variation-awareness to the HLS flow

Wang et al. recently evaluated variation-aware resources' sharing and binding algorithms[8] using Mentor Graphics' Catapult C tool on common industrial examples in wireless and image-processing applications. This work introduced two concepts, *statistical-path-delay improvement* and *criticality,* to effectively guide optimization in resource sharing and binding. The statistical path delay represents the magnitude of the path delay based on statistical analysis. Statistical path delay avoids the pessimistic nature of the worst-case path delay based on worst-case analysis. The statistical-path-delay improvement is the difference between the statistical path delay before and after resource sharing or binding.

Criticality is the probability of the operation's being on the critical delay path. This metric represents the characteristics of the path delay distribution. If an operation's criticality is small (which means that the possibility of the operation's being on the critical path is small), then the yield improvement for this particular operation won't satisfactorily contribute to overall yield improvement. The resource sharing and binding algorithm tries to minimize area under the performance yield constraint, and uses $\Delta A_C/\Delta P_I$ as the gain function to evaluate possible moves in resource sharing and binding, where A_C is area cost, and P_I is performance improvement. Our experimental results with a modified variation-aware Catapult C flow show that the average area reduction is 33%, with about 10% runtime overhead.[8]

Physical design with variation-aware HLS

As interconnect delay becomes comparable with logic delay, and because the spatial correlations of parameter variations largely affect circuits' delay and power distributions, it is becoming increasingly important to consider physical design decisions within HLS. The newly proposed layout-driven HLS framework, FastYield, integrates timing-driven floorplanning into the high-level design loop to further improve timing yield.[9] Physical information, including interconnect delay and spatial correlation on the variations of functional units, is extracted from the floorplan and used to guide the resource rebinding during the next iteration of the design loop. With the physical information, designers can more accurately estimate circuit behavior and make better design decisions during the synthesis process.

It is well known that spatial correlation of variation in parameters such as gate length can impact the variance of a circuit's timing. In the FastYield framework, Lucas et al. have proposed a unit-based correlation model in which each functional unit, register, or

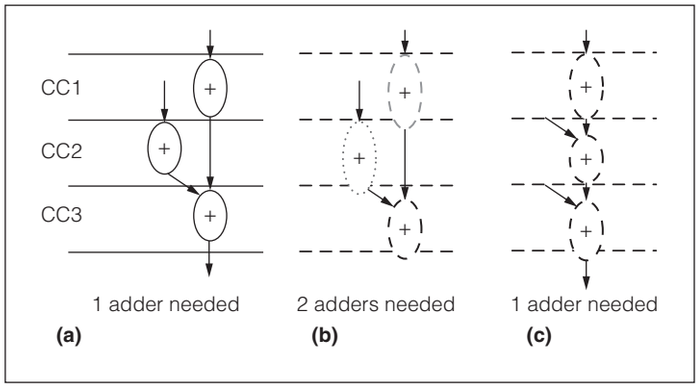

Figure 5. Potential violation on resource sharing in latch-based design: flip-flop design A (a), latch design A (b), and latch design B (c).

multiplexer is assigned a unit number and the correlation between each unit is based on the distance between the center points of the units using a correlation function.[9] After initial resource allocation and binding, a simulated annealing-based floorplanner is deployed to run the floorplanning, perform postlayout statistical timing analysis, and obtain layout information.

The method for statistical timing analysis in considering spatial correlation is based on principal component analysis (PCA), which transforms a set of correlated random variables into a new set of independent random variables. Using the unit correlation model just described, we can form a correlation matrix for the current binding solution. With PCA applied to the correlation matrix, spatial correlations of the functional units' delay variations are then captured and fed to the resource-rebinding step. In addition, the interconnect delay is modeled by the Manhattan distance and used to guide the rebinding.

By making use of accurate timing information from the design hierarchy's lower level, FastYield achieves a clock period that is 14.5% smaller, and an average performance yield gain of 78.9%, compared to a variation-unaware approach.

Register optimization

In addition to the modification of traditional HLS steps (such as resource sharing and binding), another way to mitigate delay variations in functional units is to replace edge-triggered flip-flop registers with transparent latches.[10] One unique feature for transparent latches is the time-borrowing capability. If a latch is used as the storage element between a slower unit and a faster one, timing slacks can be borrowed between consecutive control steps, thereby improving the timing yield of these two units. Consequently, an optimized register allocation technique using latches can be integrated into the yield-driven scheduling and resource-binding techniques, further improving the timing yield of designs and mitigating the impact of process variability.[10]

In dealing with latch-related circuits, latches cannot hold the output value when the input switches during the active clock period. If flip-flops are directly replaced by latches, it's possible for the circuits to have hold-time violations.[10]

As Figure 5a shows, all three `add` operations in three clock cycles, respectively, can share the same adder, because a functional unit can be reused in two consecutive control steps in flip-flop-based design. However, in Figure 5b, for latch-based design A, the operation in clock cycle 2 (CC2) cannot share the same functional unit with the operation in CC1, because the latch between CC1 and CC2 cannot store the evaluation result of CC1 stably when it tries to take a new input at CC2. This leads to a timing violation due to the resource sharing. To avoid such a violation, the operation in CC2 should be assigned to a new functional-unit instance. The operation in CC3 can still share the same unit with the operation in CC1. Consequently, two adders are needed for latch-based design A. In Figure 5c, latch-based design B with different I/O relations is shown for comparison. While all operations are chained in design B, no intermediate results need be latched. The resource sharing is therefore not affected, and only one adder is needed in the design.

To safely replace flip-flops with latches and keep the design flow compatible with traditional resource binding and sharing algorithms, Chen et al.[10] have adopted the idea of *lifetime extension*, a technique proposed by Yang et al. to eliminate the timing violations in latch-based design.[11] However, their original approach is low-power oriented, and they only changed the registers' lifetime to save power on storage elements.[11] They have not used the time-borrowing feature for latches at all; therefore, they don't need to extend the lifetime of functional units.

In contrast, the work by Chen et al.[10] relies on time borrowing between control steps, which requires a modification to the lifetime of some functional units during the HLS process. Lifetime extension involves what we call *critical operation:* If two operations in

consecutive control steps are assigned to the same functional unit in traditional HLS and the second operation does not take the output of the first operation as input, the first operation is referred to as a critical operation. For instance, the operations in steps CC1 and CC2 in Figure 5b are critical operations. To eliminate the timing and resource-sharing violations in a latch-based design, the lifetime of these critical operations should be extended by one extra cycle. The lifetime modification can be performed after the resource-binding and sharing stage in HLS. The lifetime extension of operations leads to area overhead on resource usage.

Our experimental results have shown that, on average, we can achieve a 27% performance yield improvement when latch replacement is applied, with an average area overhead of 17%.[10] Furthermore, area overhead decreases rapidly as the benchmark size increases, which indicates that the latch replacement framework is more favorable for large designs.

Postsilicon tuning and design-time optimization

To maximize design yield, designers can rely on another complementary strategy called *postsilicon tuning*, in conjunction with design-time optimization (or presilicon optimization). Postsilicon tuning approaches are performed after fabrication. Techniques such as adaptive body biasing (ABB) and adaptive supply voltage can be used to tune the fabricated chips, thus reducing the variation in delay, power, or both.[2] Postsilicon tuning can be applied at the module level and can change the modules' statistical characteristics. This tuning broadens the optimization space for the variation-aware HLS algorithms. Meanwhile, when considering the granularity of postsilicon tuning techniques, the module-level tuning is favorable because of the relatively low tuning cost.

For example, a variability-driven module selection algorithm, which Wang et al. have proposed,[12] combines design-time optimization with postsilicon tuning (using ABB) to maximize design yield. The postsilicon optimization is formulated as an efficient sequential conic program to determine the optimal body bias distribution, which in turn affects design-time module selection. With bidirectional ABB, the applied voltage can either raise the threshold voltage of the die (reverse body biasing) to reduce the leakage power at the expense of slowing down circuits or lower the threshold voltage (forward body biasing) to increase the clock frequency at the expense of higher leakage power.

ABB techniques can effectively tighten the performance and power distribution, and minimize the yield loss due to process variation. Note that the body bias voltage (also known as substrate bias voltage, V_{SB}) for each individual die is different, and therefore V_{SB} is statistically distributed according to the probability distribution of performance and power. The yield results have shown that joint design-time and postsilicon optimization can achieve on average a 28% power yield improvement, compared to the design-time-only variation-aware module selection approach.[12]

Interaction with low-power HLS

Reducing power consumption has long been a crucial design goal for researchers as technology scales. Accordingly, many HLS power optimization techniques have been proposed, including architectural transformation, idle-time maximization, and multiple supply and threshold voltages of functional units.[3] The impact of various low-power techniques on the delay and power variations during an HLS design flow has not been well studied yet, however.

For example, multi-V_{DD}/V_{TH} has been demonstrated as an effective means to reduce both dynamic and leakage power. The changes on supply and threshold voltages, however, could affect the functional units' delay and power variations.

To study the impact of different supply and threshold voltages on delay and power variations, we can characterize the components in the resource library for HLS, with a combination of different design corners. For example, Table 1 shows the delay variation characterization of a 16-bit Kogge-Stone adder and a 16-bit Brent-Kung adder, implemented in 45-nm technology.

Table 1. Delay variation (ns) characterization of two adders with dual-V_{DD} and dual-V_{TH} library.

2-adder designs		16-bit Kogge-Stone		16-bit Brent-Kung	
Case	**Power**	**Mean, μ**	**Deviation, σ**	**Mean, μ**	**Deviation, σ**
Nominal	Medium	3.26417	0.07934	3.46797	0.17013
High V_{TH}	Low	3.66011	0.09215	3.89041	0.19547
Low V_{DD}	Low	4.22523	0.11554	4.49699	0.24630

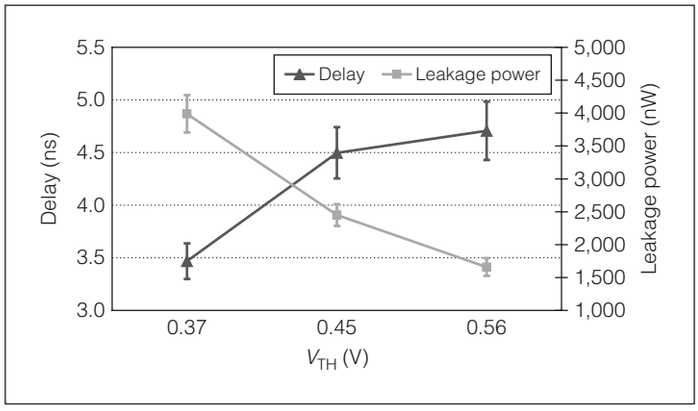

Figure 6. Delay and power trade-off with multithreshold voltage V_{TH} for a 16-bit Brent-Kung adder implemented in 45-nm technology.

The adders' delay increases with larger variations, when low-power techniques (higher V_{TH} and lower V_{DD}) are applied.

Figure 6 shows the interaction of delay variation and power variation for the 16-bit Brent-Kung adder in 45-nm technology, with different threshold voltages. It shows that the power reductions with smaller power variations are achieved at the expense of longer delay, and larger delay variation. Consequently, lower power with a more predictable power yield could result in a higher probability of timing violations. Such an observation indicates the necessity for statistical analysis and parametric yield-driven optimization approaches for low-power HLS.

Interaction with other variations

Aggressive technology scaling results not only in significant process variations but also in other spatial and temporal variations, such as temperature and aging effects, which in turn affect circuit power and performance. For example, leakage power consumption has an exponential dependency on the junction temperature. Accordingly, on-chip spatial and temporal thermal variations could have a significant impact on leakage power consumption. In addition, higher temperature causes performance degradation. Consequently, we have seen an increasing interest recently in thermal-aware HLS.[3] Current thermal-optimization techniques rely on full-chip thermal-analysis profiling, which is shown to have high computational complexity. The existence of process variations will lead to unexpected local hot-spots and further increase the complexity of the problem. Effective modeling for variation-aware thermal profiling is needed to tackle the chip thermal problem more accurately and efficiently.

Aging effects, such as electromigration, negative bias temperature instability (NBTI), and hot-carrier injection, also become more significant in the deep-submicron design regime, and they cause design parameters to change. For example, NBTI could cause the threshold voltage V_{TH} to shift during the lifetime of a design, affecting the delay and power of a functional unit.

Thus, delay and power variations due to process variation, together with the delay and power variations due to temperature variations or aging effects, increase the uncertainty of a design's performance and power estimation, and call for joint-optimization approaches such as HLS, at early design stages.

Process variation in deep-submicron VLSI design has escalated into a major challenge for designers. Dealing with delay and power variations during HLS is still in its infancy. We expect that the information in this article on statistical HLS, variation-aware resource characterization, yield-driven HLS with integrated floorplanning, register optimization, and interactions between process variations will stimulate further research work in this novel direction. ■

References

1. C. Kenyon et al., "Managing Process Variation in Intel's 45nm CMOS Technology," *Intel Technology J.,* vol. 12, no. 2, 2008; http://www.intel.com/technology/itj/2008/v12i2/3-managing/1-abstract.htm.
2. A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for VLSI: Timing and Power,* Springer, 2005.
3. P. Coussy and A. Morawiec, *High-Level Synthesis: From Algorithm to Digital Circuit,* Springer, 2008.
4. F. Wang, G. Sun, and Y. Xie, "A Variation Aware High Level Synthesis Framework," *Proc. IEEE Design, Automation and Test in Europe Conf.* (DATE 08), IEEE CS Press, 2008, pp. 1063-1068.
5. W.-L. Hung, X. Wu, and Y. Xie, "Guaranteeing Performance Yield in High-Level Synthesis," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 06), IEEE CS Press, 2006, pp. 303-309.
6. J. Jung and T. Kim, "Timing Variation-Aware High-Level Synthesis," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 07), IEEE CS Press, 2007, pp. 424-428.

7. Y. Chen, J. Ouyang, and Y. Xie, "ILP-Based Scheme for Timing Variation-Aware Scheduling and Resource Binding," *Proc. IEEE Int'l SOC Conf.,* IEEE Press, 2008, pp. 17-30.
8. F. Wang, A. Takach, and Y. Xie, "Variation-Aware Resource Sharing and Binding in Behavioral Synthesis," *Proc. Conf. Asia and South Pacific Design Automation* (ASP-DAC 09), IEEE Press, 2009, pp. 79-84.
9. G. Lucas, S. Cromar, and D. Chen, "FastYield: Variation-Aware, Layout-Driven Simultaneous Binding and Module Selection for Performance Yield Optimization," *Proc. Conf. Asia and South Pacific Design Automation* (ASP-DAC 09), IEEE Press, 2009, pp. 61-66.
10. Y. Chen and Y. Xie, "Tolerating Process Variations in High-Level Synthesis Using Transparent Latches," *Proc. Conf. Asia and South Pacific Design Automation* (ASP-DAC 09), IEEE Press, 2009, pp. 73-78.
11. W. Yang, I. Park, and C. Kyung, "Low-Power High-Level Synthesis Using Latches," *Proc. Conf. Asia and South Pacific Design Automation* (ASP-DAC 01), ACM Press, 2001, pp. 462-466.
12. F. Wang, X. Wu, and Y. Xie, "Variability-Driven Module Selection with Joint Design Time Optimization and Post-Silicon Tuning," *Proc. Conf. Asia and South Pacific Design Automation* (ASP-DAC 08), IEEE CS Press, 2008, pp. 2-9.

Yuan Xie is an associate professor in the Computer Science and Engineering Department at Pennsylvania State University. His research interests include VLSI design, electronics design automation, computer architecture, and embedded systems design. He has a PhD in electrical engineering from Princeton University.

Yibo Chen is a doctoral candidate in the Computer Science and Engineering Department at Pennsylvania State University. His research interests include statistical timing and power analysis of VLSI circuits, and variability-aware high-level synthesis. He has an MS in electrical engineering from Tsinghua University, Beijing.

■ Direct comments and questions to Yibo Chen, Dept. of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802; yxc236@cse.psu.edu.