Process-Driven Variability Analysis for Single and Multiple Voltage-Frequency Island, Latency-Constrained Systems*

Diana Marculescu, *Member*, Siddharth Garg, *Student Member*

Abstract - The problem of determining bounds for application completion times running on generic systems comprised of single or multiple voltage-frequency islands (VFIs) with arbitrary topologies is addressed in the context of manufacturing-driven variability. The approach provides an exact solution for the system-level timing yield in single clock, single voltage (SSV) and VFI systems with an underlying tree-based topology, and a tight upper bound for generic, non-tree based topologies. The results show that: (a) timing yield for overall source-to-sink completion time for generic systems can be modeled in an exact manner for both SSV and VFI systems; and (b) multiple VFI, latency-constrained systems can achieve up to 2X higher timing yield than their SSV counterparts. The results are proven formally and supported by experimental results on two embedded applications, namely software defined radio and MPEG2 encoder.

I. INTRODUCTION

Driven by aggressive technology scaling, sub-wavelength lithography is causing increased variability in process technology parameters. In addition, due to increased power density and stressed thermal envelope, system parameter variability (e.g., temperature and voltage variation) increases as well. Such process and system parameter variations can manifest themselves across a single die (within-die or WID, e.g., in the case of temperature or voltage variations) or across several dies (die-to-die or D2D, e.g., in the case of clock speed and leakage power variations). In addition, variations can be systematic or random, static or dynamic in nature.

Irrespective of their source or manifestation, variability poses a major challenge in designing complex single chip systems. Due to increased parameter variation, reliability of logic and memory available on chip decreases significantly. Design methodologies and tools that take into account these variations are needed for all levels of abstraction present in current design flows. While a significant body of work exists for characterizing performance and power consumption in the presence of process-driven variability at low levels of abstraction (i.e., the interface between physical-gate levels) [1]-[4], models of these effects need to be provided at higher

levels of abstraction as well. Current high-level design methodologies and tools (namely targeting register transfer - RT - and system levels) still assume a classic *static timing* behavior and do not include effects of variability on performance or energy. In such cases, design optimization tools or exploration frameworks are likely to provide suboptimal solutions or designs that might not satisfy given requirements in the presence of severe variability. In support of a complete probabilistic design flow, high-level modeling of variability effects is needed for determining design choices that are *most likely* to meet initial design constraints.

The work presented in this paper extends in a comprehensive manner our initial analysis on system-level variability analysis [5] and provides a starting point in this direction, by enabling high-level analysis of variability on overall system performance and its interactions or trade-offs with energy. As a design driver for our framework we consider the case of complex systems on chip (SoCs) comprised of a set of intellectual property (IP) cores or processing elements. Recently, design methodologies based on voltage-frequency islands (VFI) have been proposed [6]-[8] as a possible solution to energy efficiency by allowing each island to run at its own voltage (speed) such that performance constraints are met, while energy is minimized. Techniques that allow either static or dynamic voltage/speed assignment [6][8] for each island have been proposed and shown to increase energy efficiency significantly, under given latency or rate constraints. Although implicitly assumed many times, the benefits of using multiple clock or voltage islands on variability effects have not been proven so far. This paper provides a *formal proof* showing that VFI-based systems are *more likely* to meet timing constraints than their single clock, single voltage (SSV) counterparts, under most general assumptions, that is generic distributions characterizing local VFI clock cycle times. Furthermore, this paper describes practical ways of characterizing overall system variability via *stochastic upper bounds* for the end-to-end system latency or completion time via *stochastic ordering* [9]-[12] and simple combinations of multiplication and/or convolution of individual component distributions.

II. RELATED WORK AND PAPER CONTRIBUTION

This paper addresses the problem of determining the timing yield for latency-constrained systems with an underlying *synchronous, single voltage* (SSV) or *voltage-frequency island-based* (VFI) implementation. While statistical timing and power analysis including process-driven variability effects has become a hot area of research recently, the problem has only been addressed at logic/gate levels of abstraction. Statistical timing analysis has been addressed at logic level for the case of generalized distributions [1], as well as in the case of considering arbitrary correlations and

Manuscript received March 14, 2007, revised June 30, 2007, October 22, 2007, and accepted October 26, 2007. This research was supported in part by Semiconductor Research Corporation under contract no. 2005-HJ-1314.

The authors are with Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: dianam@ece.cmu.edu; sgarg1@ece.cmu.edu).

Copyright (c) 2008 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubpermissions@ieee.org.

stochastic bounds [2]. Fast heuristics have been proposed for both statistical timing analysis [3] as well as variability aware leakage power estimation [4]. Commercial products have recently started to address power optimization and management by including voltage islands in their design flows, but they do not provide performance yield analysis at the level of abstraction assumed by us. For example, Magma’s BlastPower is a *back-end* tool targeting power optimization through the use of voltage islands at physical design/circuit levels. However, it does not directly address the same concerns as our work which is a *front-end* approach targeting performance yield analysis at system level for VFI and SSV systems.

While this is by no means an exhaustive enumeration of existing work in this area, they all address the variability problem at logic/gate level, and do not provide capabilities for composability of these per core (or per module) models for whole system-level analysis of process-driven variability impact. With design refinement starting at higher and higher levels, there is an increased (yet, so far unsupported) need for system-level characterization of variability effects on both performance and power consumption.

While there exist statistical timing analysis techniques developed for gate/circuit level descriptions, there are several features that make our approach distinct than existing frameworks:

- Many statistical timing analysis techniques assume normal delay distributions for gates, whereas our approach is general and does not assume anything about the per core distribution.
- Techniques that do assume generic distributions (e.g., [1]) cannot be applied in the case of SSV systems: while at logic level, gate delays are unconstrained (except for correlations induced by the process), in the case of SSV systems, *all* cores are clocked by the same global clock which makes the per core latencies multiples of the *same global cycle time*. Existing statistical timing analysis techniques do not handle this type of constraint.
- Furthermore, due to its generality and applicability to both SSV and VFI systems, the approach can be applied to VFI systems of various granularities, allowing for analysis of performance yield for a given system under various number of VFIs. We present one such example in our experimental results in Section VI.

So far, at system level, performance and power analysis largely relies on “perfect” hardware assumption, that does not include any uncertainties induced by the manufacturing process or dynamic parameter variations. The problem of performance analysis for systems with global asynchronous communication has been addressed before [13], but neither cases considered process variability impact on overall timing yield or latency variability.

The main contribution of the paper is to provide a first step in the direction of analyzing statistical properties at system level in the presence of variations. Specifically, the approach presented in this paper:

- Provides a *general approach* for characterizing *global system level timing yield* for complex systems, relying on generic distributions characterizing individual modules.
- Shows that VFI-based latency-constrained systems are always *more likely* to meet given timing constraints than their SSV counterparts under the same communication latency assumption.

The results presented in this paper can be useful for system developers that need to consider driver applications characterized by real-time latency constraints and thus, need to determine upfront the best design choice for meeting these constraints. Tools

Figure 1. (a) A SSV-based component graph with cores (PEs) clocked by a single global clock signal and characterized by a global voltage level; (b) A VFI-based component graph as in [6] with cores (PEs) characterized by local speeds/voltages.

supporting this decision as early in the design process as possible are desirable and become mandatory with increased design complexity and variability effects.

III. PRELIMINARIES AND ASSUMPTIONS

Without loss of generality, we consider the case of systems comprised of a number of synchronous cores, IPs or processing elements (PEs)¹ (homogeneous or heterogeneous). We consider both the case of fully *synchronous/single voltage* (SSV) systems, as well as the case of systems based on *voltage-frequency islands* (VFIs). In the latter case, PEs can be assigned to a single VFI (in other words, cores cannot belong to more than one VFI).

A VFI might consist of a single PE or, depending on the physical or design considerations, may include a group of PEs. We assume that power in the case of both SSV and VFI systems is supplied by an off- or on-chip source and can be controlled independently for a VFI. This may be achieved by using either on-chip voltage regulators or multiple power grids [14]. Since each VFI is locally synchronous, it is assumed to be clocked using a ring oscillator controlled by the intra-island supply voltage using a digital phased lock loop [15][16]. In the case of SSV systems, communication is synchronous, point-to-point, while in the case of VFI systems, it is implemented via a modified version of mixed-clock FIFOs [17] that also allows for voltage level conversion.

In both SSV and VFI cases, we assume that the allocation and mapping of various processes or computational kernels of the application to PEs, as well as the number and types of the communication links and PEs have already been determined. We also assume that the processes have already been scheduled on their

1. Cores, IPs or PEs will be used interchangeably throughout the paper.

respective processing elements. For VFI systems, a bounded number of storage cells is available in the mixed-clock FIFOs used between two communicating PEs.

To this end, the system comprised of communication cores is modeled using a *component graph*. In a component graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, cores are modeled as communicating processes (nodes) that have associated communication channels between them (edges). As shown in Figure 1., in the case of SSV systems, all cores/PEs are clocked globally (thus being characterized by *correlated* latencies via the common cycle time), while in the case of VFI systems, each core/PE is part of a single voltage-frequency island, as noted before (thus being characterized by *independently distributed* latencies, as local clocks are not correlated).

This paper addresses the impact that *WID random process variability* has on performance for applications implemented as systems on a chip, using either a SSV or VFI organization. The reason we only consider WID variations and not D2D variations is that D2D variations basically determine the *variance* of the cycle time, while its *mean* is mainly determined by the WID variations (as also pointed out by Bowman et al. [19]). Since one of the goals of this paper is to compare VFI and SSV systems in terms of performance under process variability and both VFI and SSV systems are affected the same way by D2D variations, we restrict ourselves to WID variations only. In addition, we address the impact of WID *random* variations only. We note that the proposed framework can model *systematic* WID variations quite easily, since systematic variations translate into a *shift in the mean of the distribution*. In fact, the results for the VFI implementation will be even better when compared with the SSV implementation if systematic variations are included: systematic WID variations further heighten differences between the frequencies of various parts of the die. For example, the corners of the die may run slower than the regions in the middle. In this case, SSV designs will need to run at the slowest speed on the entire die, while PEs in the center of the die for VFI designs can run faster than PEs on the sides. Such systematic variations where corners run slower than the middle portion of the die has indeed been observed before [20]. Therefore, we chose to be conservative and model only the (hard to model) random WID variations.

Since variability affects timing yield, natural *design drivers* for the considered problem are applications characterized by *real time deadline constraints*. So far, such applications have only been analyzed from a *worst-case* timing perspective, by using static timing assumptions, without relying on process-induced probabilistic or statistical models for their timing behavior.

IV. THEORETICAL FORMULATION

Consider the component graph in Figure 1. To this end, we will assume the following, without loss of generality:

- The component graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is characterized by the set of nodes $\mathcal{V} = \{1, 2, \dots, n\}$ and edges $\mathcal{E} = \{(i, j) \mid i \text{ precedes } j\}$.
- Although the underlying component graph model may include feedback paths, in the theoretical treatment we restrict ourselves to *directed acyclic graphs* (DAGs). General graphs have been shown to be reducible to acyclic component graphs by lumping strongly connected components (SCCs) including feedback loops into *supernodes* [6],[18]. As shown in [18], the processing rates of these supernodes (and thus, their latencies in cycle counts) can be found by averaging across all nodes in the SCC.

- The component graph includes a single source node (s) and a single sink node (S). Graphs including multiple sinks or source nodes can be reduced to this case by adding dummy, zero-latency source (sink) nodes feeding into (from) the actual source (sink) nodes.
- Each node i ($1 \leq i \leq n$) in the component graph is characterized by the number of cycles¹ C_i it takes to process one item of data (be it, packet - as in the case of networking applications, or macroblock/ frame - as in the case of video streaming applications, for example). While in the most general case, C_i 's vary depending on the workload and type of data processed, we restrict ourselves to considering C_i a *constant* characterizing per core *typical* or *worst-case* cycle-count and not a time-varying, workload-dependent variable. This way, we separate the effect of *technology-induced variability* from the effect of *workload-induced variability*. While there have been significant research efforts in the past to perform static analysis on task graphs using average or worst case number of execution cycles (e.g., [21]), our goal is to analyze the impact of process variations in cycle time affecting system end-to-end latency, which is orthogonal, from an analysis perspective, to the number of cycles taken by an application running on a given platform. We note that C_i is *application* and *platform* dependent, while the effect of technology and microarchitectural characteristics is included in the actual distribution of the cycle time for node i (T_i).
- If manufacturing process-induced variations are considered, the latency for core i can be modeled as $L_i = C_i \cdot T_i$ where cycle time of core i (T_i) is characterized by a *probability density function* (*pdf*) which includes effects not only from the process, but also from the underlying architecture (via the effective number of independent critical paths). Our work assumes that each IP or core i is pre-characterized in terms of its cycle time distribution. Such a distribution can be obtained via existing logic/circuit level techniques applied for each core [1]-[3] or it can rely on the generic *critical path model* described by Bowman et al. [19]. Indeed, as shown in the FMAX model proposed and validated by Intel [19], the cycle time of a complex design can be characterized by a *pdf* that is a function of the single gate delay and the effective number of independent critical paths. Thus, our approach can work with any type of distribution, be it the result of a lower level statistical timing analysis technique or a closed-form representation, such as the one in [19]. Thus, core i latency L_i is also characterized by a *pdf* $f_i(t)$:

$$f_i : \mathbb{R} \rightarrow [0, \infty) \quad (1)$$

and *cumulative distribution function* (*cdf*):

$$Pr(L_i \leq t) = F_i(t) = \int_{-\infty}^t f_i(u) du \quad (2)$$

The *expectation* of each random variable L_i can thus be defined as:

$$E[L_i] = \int_{-\infty}^{\infty} t f_i(t) dt = \int_0^{\infty} [1 - F_i(t)] dt \quad (3)$$

1. Since this paper does not target computation/communication analysis or trade-offs, the communication latency is assumed to be lumped in C_i . The difference between SSV and VFI systems communication latency modeling will become apparent in the sequel.

since $F_i(t)$ is a positively defined function ($F_i(t) = 0$ for $t < 0$).

- For each node i , we define its *completion time* Λ_i as the source-to-node i latency: $\Lambda_i = \max_{j \in \text{pred}(i)} \{L_j\} + L_i$ where $\text{pred}(i)$ is the set of all nodes directly preceding node i (empty set if i is a source node). Λ_i is characterized by distribution $\Phi_i(t)$ and the completion time for sink node S is the overall graph completion time: $L = \Lambda_S$ with distribution $\Pr(L \leq t) = F(t) = \Phi_S(t)$. For example, in Figure 1, the completion time for the component graphs shown is the source-to-sink latency $L = \Lambda_S$.
- Completion times for nodes are assumed to be *structurally correlated* only. In other words, only correlations due to underlying topology are considered and thus *pdfs* f_i and *cdfs* F_i are assumed to be independent for all nodes i . In general, process-driven variability inadvertently introduces *spatial* correlations, in addition to existing *structural* correlations induced by the interconnect network topology. While spatial correlations between process parameters may lead to some degree of correlation between the IP timing random variables, the magnitude of correlations seen between IP cycle times is typically much smaller than between individual gates in the circuit during traditional statistical timing analysis, since spatial correlations tend to die out quickly as a function of distance (for example, [22] showed that spatial correlations go to zero at a distance of half the die length). Related work at gate-level that has considered spatial correlations (e.g., [2]-[3]) also determined that only adjacent gates are highly correlated - this translates to negligible correlations between large IPs, like the ones considered in our work. Furthermore, while gate length has been shown to possess spatial correlation characteristics, an important source of threshold voltage variation is *random dopant fluctuation* (RDF) that is purely random from transistor to transistor - this random component tends to further decrease the correlation between IP cycle times. Since the level of abstraction at which we address this problem is much higher than existing work (i.e., gate-level), at this coarser level of granularity, it is expected that structural correlations due to system topology will be dominating, while spatial correlations can be considered negligible for all practical purposes.

The goal of this work is twofold:

- (i) Given the system component graph and associated node distributions, find the resulting distribution for the overall system completion time L ($F(t)$), and thus the likelihood that a given latency constrained system will satisfy a given timing constraint, under process-driven variations.
- (ii) Characterize formally and quantitatively the relation between completion time of VFI-based systems and their SSV counterparts, thus allowing system designers to choose one organization versus the other early in the design process depending on their likelihood of meeting required timing constraints.

Specifically, while existing work in statistical timing analysis has addressed the problem of finding the timing yield for circuits (or cores) of moderate complexity, the same problem for complex systems comprised of interconnected IPs or cores has not been addressed. Assuming that f_i (F_i) have already been determined by a detailed analysis applied to each core or PE, this paper provides a framework for characterizing overall system variability, be it SSV

or VFI based, their relationship in terms of timing yield, as well as its impact on system level performance trade-offs.

IV.A. Stochastic Bounds for Completion Time

Since distributions f_i might be general and may not be reducible to well-known forms (e.g., Gaussian), to address the problem of determining *cdf* F in (3) above, we rely on the concept of stochastic bounding which has been used before, but in the context of statistical timing analysis at lower levels of abstraction (i.e., gate-level) [2]. Since we assume that *spatial* correlations can be neglected at this level of abstraction (as described before) we will rely only on including the impact of *structural* correlations due to the inter-core communication topology. For determining bounds on graph completion time, we use the concept of *stochastic ordering* [23] and *positively associated variables* [24].

Definition 1. [23] Given two random variables X and Y with positively defined *cdfs* F_X and F_Y , X is *stochastically smaller* than Y (written as $X \leq_{st} Y$ or $F_X \leq_{st} F_Y$) if $F_X(u) \geq F_Y(u)$ ¹ for every $u \geq 0$.

Definition 2. [24] The variables of random vector $X = \{X_1, X_2, \dots, X_n\}$ are *positively associated* if for all monotone, non-decreasing functions $f, g: \Re^n \rightarrow \Re$ for which expectations exist, $\text{Cov}[f(X), g(X)] \geq 0$ where the *covariance matrix* (*Cov*) between random vectors X and Y is defined as $\text{Cov}[X, Y] = E[(X - E[X]) \cdot (Y - E[Y])^T]$.

It has been shown [12] that stochastic ordering can be used to characterize upper bounds for positively associated random variables. Indeed, if variables of random vector $\bar{X} = \{\bar{X}_1, \bar{X}_2, \dots, \bar{X}_n\}$ are an independent version of $X = \{X_1, X_2, \dots, X_n\}$ (i.e., $\{\bar{X}_1, \bar{X}_2, \dots, \bar{X}_n\}$ are mutually independent and $X_i, \bar{X}_i$ have the same probability distribution), then $X \leq_{st} \bar{X}$, and thus $\bar{X}$ is an upper bound for X .

Stochastic ordering is preserved across multiplication and convolution of distributions [23], i.e., if two distribution families $(F_1, F_2, \dots, F_n)$ and $(G_1, G_2, \dots, G_n)$ satisfy $F_i \leq_{st} G_i$ for all $1 \leq i \leq n$, then:

$$\prod_{i=1, \dots, n} F_i \leq_{st} \prod_{i=1, \dots, n} G_i \quad (4)$$

$$\bigotimes_{i=1, \dots, n} F_i \leq_{st} \bigotimes_{i=1, \dots, n} G_i \quad (5)$$

where $\otimes$ is the convolution operator defined for *cdfs* F_1 and F_2 as:

$$(F_1 \otimes F_2)(t) = \int_{-\infty}^t \int_{-\infty}^{\infty} f_1(u) f_2(v-u) du dv \text{ where } f_1 (f_2) \text{ is the pdf}$$

corresponding to F_1 (F_2).

If random variables above are path or node latencies in a component graph, note that the *product* distribution corresponds to a *parallel* combination of nodes or paths (*all* of them need to satisfy timing constraints), while *convolution* corresponds to a *series* combination (their *sum* needs to satisfy given timing constraints). For example, as shown in Figure 2., convolution is used to

1. Note that we write $F_X \leq_{st} F_Y$, even if $F_X(u) \geq F_Y(u)$.

determine path latency distributions, which are then used to determine source-to-sink latency distribution via path *cdf* multiplication.

Existing work in determining stochastic bounds for graph completion times [9]-[12],[25] have all relied on building new graphs for which computing completion times is easy to perform via product or convolution operations, such as the case of series-parallel DAGs. These newly built graphs are characterized by completion times that *stochastically bound* the initial graph completion time, but are easier to compute since they rely only on applying equations (4) and (5) above for *parallel* and *series* combinations, respectively. To determine upper or lower bound series-parallel combinations for a given initial graph, simple node/arc addition or removal is used. Intuitively, insertion of edges (arcs) in the original graph increases dependencies, thus leading to an *upper bound* on execution time, while deletion of arcs causes less dependencies and smaller execution times (hence a *lower bound*). Furthermore, path latencies in the new graphs are random variables that form an *independent version* of the initial positively associated latencies, and thus provide stochastic bounds for overall completion time.

$$Pr(L_{1-2-6} \leq t) = (F_1 \otimes F_2 \otimes F_6)(t)$$

Figure 2. A series-parallel component graph and source-to-sink latency computation for given node latency distributions.

We show in Figure 3. some examples of how an initial graph can be reduced to a series-parallel combination. Figure 3.(a) shows a simple graph that is known as a *forbidden graph* [10] as it does not correspond to a simple series-parallel combination. However, by duplicating nodes 2 or 3, one can obtain three series-parallel versions that have been shown to provide upper bounds for the original graph completion times (Figure 3.(b)-(d)). The problem of determining *which* of these series-parallel versions provides the tightest upper bound is an NP-complete problem, hence most existing work in this area relies on heuristics [9]-[12] and uses non-deterministic choices to determine a possible associated series-parallel graph.

One of the *deterministic* algorithms that can be automated and applied for generic component graphs is due to Shogan and relies on duplicating nodes so as to obtain a *tree graph* with paths that *stochastically upper bound* completion times of the original graph. As shown in Figure 3.(e), the original graph G is transformed into a simple tree graph $\bar{G}$ whose completion time stochastically bounds the initial graph completion time. Furthermore, since the approach is deterministic in manner, the achieved bound is tighter than other heuristics that try to find series-parallel upper bound graphs. The result has been proven formally in [25]:

Lemma 1. [12][25]¹ Consider a component graph $G(\mathcal{V}, \mathcal{E})$.

Stochastic upper bounds for node and overall graph completion time can be found from its *associated graph* $\bar{G}(\mathcal{V}, \mathcal{E})$ defined as follows:

(i) If i is a source node in $G(\mathcal{V}, \mathcal{E})$, let $\bar{G}_i$ be the graph containing the single node i . The *cdf* for completion time of node i in $\bar{G}_i$ is defined as $\bar{\Phi}_i(t) = \Phi_i(t) = F_i(t)$.

(ii) If i is not a source node, let $\bar{G}_{pred(i)}$ be the graph obtained by considering all $\bar{G}_j$ in parallel, where j is a direct predecessor of i in the original component graph. Thus, $\bar{G}_i$ is defined as the graph connecting $\bar{G}_{pred(i)}$ in series with node i , with *cdf* for node i

$$\text{defined as } \bar{\Phi}_i(t) = \left[\prod_{j \in pred(i)} \bar{\Phi}_j(t) \right] \otimes F_i(t) \leq \Phi_i(t). \quad (6)$$

(iii) If j are all sink nodes in the original graph, $\bar{G}$ is obtained by considering all $\bar{G}_j$ in parallel (connected in series with a dummy node if originally there is more than a sink node). In this case, the *cdf* for the overall completion time of graph $\bar{G}$ is given by

$$\bar{\Phi}(t) = \prod_{j \text{ sink node}} \bar{\Phi}_j(t) \leq F(t) \text{ which provides a stochastic upper bound for the original graph completion time: } \Lambda_{\bar{G}} \geq_{st} \Lambda_G. \square$$

Figure 3. (a) An irreducible graph (forbidden graph [10]); (b)-(d) Three completely reducible forms (series-parallel combinations) for the graph in (a); (e) An example of reducing a graph G to its tree-like counterpart $\bar{G}$ by node duplication [12].

We show in Figure 3.(e) an example in which Shogan's algorithm has been applied using Lemma 1 above. As it can be seen, in the associated graph all source-to-sink paths have been duplicated such that they are spatially independent as they do not share any nodes. According to Lemma 1, the associated graph $\bar{G}$ has a completion

1. The proof can be found in [12].

time that upper bounds (in stochastic sense) the completion time of the original graph $\mathcal{G}$

IV.B. Completion Time in VFI vs. SSV Systems

The theoretical formulation presented in Section IV.A. provides a way of bounding execution time for systems comprised of cores or IPs characterized by independent, generic distributions. More precisely, the formulation makes a very important assumption, that is, there are no other constraints among execution times of various cores. However, in SSV systems this assumption does not hold true. Consider, for example, the case of SSV from Figure 1. - in this case, the latency for core i is $L_i = C_i \cdot T$, where T is the global clock cycle time (the same for all cores). For given cycle counts per core (typically obtained by worst-case or typical case analysis or profiling), per core latencies are *constrained* and related to each other as follows: $L_i/C_i = T$. Under such assumptions, the probability that a given path p_j with latency π_j in the component graph satisfies a timing constraint t can be expressed as:

$$Pr(\pi_j \leq t) = Pr\left[\bigcap_{i \in p_j, \sum t_i = t, t_i/t_k = C_i/C_k} (L_i \leq t_i)\right] \quad (7)$$

which is *different* than the same probability in the VFI case where cores are *not constrained* by a global clock speed:

$$Pr(\pi_j \leq t) = Pr\left[\bigcap_{i \in p_j, \sum t_i = t} (L_i \leq t_i)\right] \quad (8)$$

While for equation (8) variables t_i 's are not restricted to be multiples of a single global clock cycle time and a true convolution is applicable for determining the overall probability, the same is not true for equation (7). Furthermore, we will show formally that the probability in (7) is smaller than the one in (8), thus making SSV systems more likely to not meet timing constraints, when compared to their VFI counterparts.

To be able to compare the two cases, one might wonder if an equivalent algorithm for determining stochastic upper bounds exists for SSV systems. We show that this is indeed possible:

Lemma 2. Consider a component graph of an SSV system $\mathcal{G}_{SSV}(\mathcal{V}, \mathcal{E})$. Stochastic upper bounds for node and overall graph completion time can be found from its *associated graph* $\bar{\mathcal{G}}_{SSV}(\bar{\mathcal{V}}, \bar{\mathcal{E}})$ defined as in Lemma 1, except for equation (6) which is replaced by:

$$\bar{\Phi}_i(t) = \left[\prod_{j \in pred(i)} \bar{\Phi}_j(\alpha_i t) \right] F_i((1 - \alpha_i)t) \leq \Phi_i(t) \quad (9)$$

where $\alpha_i = (max_j C_j) / (max_j C_j + C_i)$.

Proof: For SSV systems, a series combination of two nodes has the overall *cdf* $F_1(\alpha t)F_2((1 - \alpha)t)$ where $\alpha = C_1 / (C_1 + C_2)$ since latencies for both nodes are constrained by the common cycle time T : $L_1/C_1 = L_2/C_2 = T$ and thus $L_1/L_2 = C_1/C_2$. Since the approach described in Lemma 1 (and also used here) replaces overlapping paths with non-overlapping, *independent* counterparts by using duplicate (independent) nodes characterized by the same probability distributions as the original nodes, according to stochastic bounds for positively associated independent random variables [12], overall path latencies in the tree graph $\bar{\mathcal{G}}_{SSV}$ are stochastic upper bounds for the path latencies in the original graph

$\mathcal{G}_{SSV}$. $\square$

We note that the results provided by Lemma 1 and Lemma 2 can be used standalone for VFI and SSV systems, or in a mix-and-match manner in the case of clustered VFI systems with more than one core per VFI. More precisely, Lemma 1 can be used to determine the completion time for each synchronous cluster or VFI, and the results combined and used by Lemma 2 to determine the overall completion time for the clustered VFI system. We show the results of such an approach in the experimental results of Section VI.

Based on Lemma 1 and Lemma 2, we are now able to prove that SSV systems are less likely to meet timing constraints than their VFI counterparts. The result is proved in the following two theorems.

Theorem 1. Consider the distribution (*cdf*) family $(F_1, F_2, \dots, F_n)$ and associated *pdfs* $(f_1, f_2, \dots, f_n)$, $F_i, f_i : \mathbb{R} \rightarrow [0, \infty)$. Let $\{\alpha_i\}$ be a set of fixed non-negative numbers with $0 \leq \alpha_i \leq 1$ and

$$\sum_i \alpha_i = 1. \text{ Then } \bigotimes_{i=1, \dots, n} F_i(t) \geq \prod_{i=1, \dots, n} F_i(\alpha_i \cdot t).$$

Proof: The proof is by induction on n .

Initial step: We show first that the inequality holds for $n = 2$ and a fixed, given value $0 \leq \alpha \leq 1$: $(F_1 \otimes F_2)(t) \geq F_1(\alpha t)F_2((1 - \alpha)t)$

for any $t \geq 0$. According to the definition of convolution and *cdf*:

$$(F_1 \otimes F_2)(t) = \int_{-\infty}^t \int_{-\infty}^{\infty} f_1(u)f_2(v - u)du dv \quad (10)$$

$$F_1(\alpha t)F_2((1 - \alpha)t) = \left(\int_{-\infty}^{\alpha t} f_1(u)du \right) \left(\int_{-\infty}^{(1 - \alpha)t} f_2(u)du \right) \quad (11)$$

Equation (10) can be re-written as:

$$\begin{aligned} \int_{-\infty}^t f_1(u) \left(\int_{-\infty}^t f_2(v - u)dv \right) du &= \int_{-\infty}^t f_1(u)F_2(t - u)du \\ &\geq \int_{-\infty}^{\alpha t} f_1(u)F_2(t - u)du \geq F_2(t - \alpha t) \int_{-\infty}^{\alpha t} f_1(u)du \end{aligned} \quad (12)$$

since $f_1(u)F_2(t - u)$ is always positive and F_2 is monotonically increasing and thus $F_2(t - u) \geq F_2(t - \alpha t)$ for any $u \leq \alpha t$. From (12) and (10), the claim of the theorem follows immediately.

Inductive step: Assuming that the inequality holds for k , we show that it also holds for $k + 1$ ($k \geq 2$). Consider $\{\alpha_i\}_{1 \leq i \leq k+1}$ with

$0 \leq \alpha_i \leq 1$ such that $\sum_{i=1}^{k+1} \alpha_i = 1$. Let $\beta_j = \alpha_j / \sum_{i=1}^k \alpha_i$ for $j = 1, \dots, k$. Then:

$$\bigotimes_{i=1, \dots, k+1} F_i(t) \geq (\Phi \otimes F_{k+1})(t) \text{ where} \quad (13)$$

$$\Phi(t) = \prod_{i=1, \dots, k} F_i(\beta_i \cdot t) \quad (14)$$

since stochastic inequalities are preserved across convolution and the inequality stated in the theorem holds for $k \geq 2$ and fixed constants $\{\beta_i\}_{1 \leq i \leq k}$. Applying the inequality for functions F_{k+1} ,

Φ , and constants α_{k+1} and $(1 - \alpha_{k+1}) = \sum_{i=1}^k \alpha_i$, we get:

$$(\Phi \otimes F_{k+1})(t) \geq \Phi((1 - \alpha_{k+1})t)F_{k+1}(\alpha_{k+1}t) \quad (15)$$

Based on the definition of Φ in (14), we get:

$$\begin{aligned}\Phi((1 - \alpha_{k+1})t) &= \prod_{i=1, \dots, k} F_i(\beta_i \cdot (1 - \alpha_{k+1}) \cdot t) \\ &= \prod_{i=1, \dots, k} F_i(\alpha_i \cdot t)\end{aligned}$$

and from (13) and (15), the claim of the theorem follows. $\square$

Theorem 1 can be used to formally show that SSV are less likely to meet timing constraints under process-driven variations than their VFI counterparts in the case of series-parallel (and thus, tree) topologies. The result is proved by the following theorem.

Theorem 2. Consider the SSV and VFI systems as in Figure 1. Assuming that the communication latency is the same for both SSV and VFI systems, for the case of series-parallel topologies, if L_{SSV} and L_{VFI} are their corresponding completion times, then:

$$Pr(L_{SSV} \leq t) = F_{SSV}(t) \leq F_{VFI}(t) = Pr(L_{VFI} \leq t) \quad (16)$$

for any $t \geq 0$. In other words $L_{SSV} \geq_s L_{VFI}$ and thus, the SSV system is *less likely* to meet timing constraints than its corresponding VFI counterpart.

Proof: (Sketch) As described in Lemmas 1 and 2, completion time can be easily computed for component graphs with a tree-like topology. Theorem 1 basically shows that path (or series combination) latencies in SSV systems are stochastically larger than the ones in VFI systems. Since (according to (4)) stochastic ordering is preserved across multiplication, VFI systems are better (stochastically) than SSV systems for *parallel* combinations as well. More formally, since latency π_j for any path p_j in a SSV or VFI implementation satisfies $\pi_{j, VFI} \leq_s \pi_{j, SSV}$ according to Theorem 1, then for any series-parallel combination of independent paths $\{p_j\}$, we have:

$$\begin{aligned}Pr\left(\bigcap_j (\pi_{j, SSV} \leq t)\right) &= \prod_j Pr(\pi_{j, SSV} \leq t) \\ &= \prod_j F_{j, SSV}(t) \leq \prod_j F_{j, VFI}(t) = Pr\left(\bigcap_j (\pi_{j, VFI} \leq t)\right)\end{aligned}$$

and thus any series-parallel combination of nodes in a component graph preserves the stochastic order between SSV and VFI. $\square$

The theoretical results shown in this section can be used to determine how much more likely is a VFI system to satisfy given timing constraints than a SSV system. For a given required timing yield, the gap between the two completion times can be used to check whether additional latency due to inter-island communication will reduce this gap and if any leftover difference between completion times can be traded-off for energy efficiency. More specifically, consider completion times $L_{SSV, y}$, $L_{VFI, y}$ for timing yield y (i.e., completion times for which probability is exactly y : $F_{SSV}(L_{SSV, y}) = y$, $F_{VFI}(L_{VFI, y}) = y$). The VFI system can tolerate any synchronization penalty L_s that satisfies $L_s \leq L_{SSV, y} - L_{VFI, y}$, and can use any leftover slack $L_{SSV, y} - L_{VFI, y} - L_s$ for further reducing energy, while maintaining the same performance as the original SSV system by assigning voltages to local VFIs, under given latency constraints. Recent work [6] has shown that this is possible for VFI systems with significant reduction in overall energy requirements.

V. PRACTICAL CONSIDERATIONS

The results from the previous section are based on a continuous distribution for PE/core or overall system latencies. The assumption that core cycle times (and thus, latencies) can take any value in a continuous domain may not be valid from a practical point of view. In this section, we show that the same theoretical results can be found in the case of discrete distribution assumption. In this case, core i latency L_i is characterized by a *probability mass function* (pmf) $f_i(t)$, $f_i: \mathbb{N} \rightarrow [0, \infty)$ and *cumulative distribution function* (cdf): $Pr(L_i \leq t) = F_i(t) = \sum_{u=-\infty}^t f_i(u)$.

As shown in the sequel, Lemmas 1 and 2 are directly applicable to discrete distributions as well, since stochastic ordering results are preserved in the case of discrete distributions. We will prove in the following that Theorem 1 also holds for discrete distributions, and from here, the result that VFI systems are stochastically better than SSV systems in the discrete case as well.

Theorem 3. Consider the discrete distribution (cdf) family $(F_1, F_2, \dots, F_n)$ and associated pmfs $(f_1, f_2, \dots, f_n)$, $F_i, f_i: \mathbb{N} \rightarrow [0, \infty)$. Let $\{t_i\}$ be a set of fixed non-negative latencies with $0 \leq t_i \leq t$ and $\sum_i t_i = t$ for a given $t \in \mathbb{N}$. Then

$$\bigotimes_{i=1, \dots, n} F_i(t) \geq \prod_{i=1, \dots, n} F_i(t_i).$$

Proof: The proof is by induction on n .

Initial step: We show first that the inequality holds for $n = 2$ and fixed, given values $0 \leq t_1, t_2 \leq t$:

$$(F_1 \otimes F_2)(t) \geq F_1(t_1)F_2(t_2) \quad \text{for any } t \geq 0 \text{ and } t_1 + t_2 = t.$$

According to the definition of convolution and cdf:

$$(F_1 \otimes F_2)(t) = \sum_{v=-\infty}^t \sum_{u=-\infty}^{\infty} f_1(u)f_2(v-u) \quad (17)$$

$$F_1(t_1)F_2(t_2) = \left(\sum_{u=-\infty}^{t_1} f_1(u)\right) \left(\sum_{u=-\infty}^{t_2} f_2(u)\right) \quad (18)$$

Equation (17) can be re-written as:

$$\begin{aligned}\sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^t f_1(u)f_2(v-u) &= \sum_{u=-\infty}^{\infty} f_1(u)F_2(t-u) \\ &\geq \sum_{u=-\infty}^{t_1} f_1(u)F_2(t-u) \geq F_2(t-t_1) \sum_{u=-\infty}^{t_1} f_1(u)\end{aligned} \quad (19)$$

since $f_1(u)F_2(t-u)$ is always positive and F_2 is monotonically increasing and thus $F_2(t-u) \geq F_2(t-t_1)$ for any $u \leq t_1$. From (19) and (17), the claim of the theorem follows immediately.

Inductive step: Assuming that the inequality holds for k , we show that it also holds for $k+1$ ($k \geq 2$). Consider $\{t_i\}_{1 \leq i \leq k+1}$ with

$0 \leq t_i \leq t$ such that $\sum_{i=1}^{k+1} t_i = t$. Then:

$$\bigotimes_{i=1, \dots, k+1} F_i(t) \geq (\Phi \otimes F_{k+1})(t) \quad \text{where} \quad (20)$$

$$\Phi(s) = \prod_{i=1, \dots, k} F_i(t_i) \quad \text{if } s = \sum_{i=1}^k t_i \quad (21)$$

since stochastic inequalities are preserved across convolution and

the inequality stated in the theorem holds for $k \geq 2$ and fixed integers $\{t_i\}_{1 \leq i \leq k}$. Applying the inequality for functions F_{k+1} , Φ , and integers t_{k+1} and $s = (t - t_{k+1}) = \sum_{i=1}^k t_i$, we get:

$$(\Phi \otimes F_{k+1})(t) \geq \Phi(t - t_{k+1})F_{k+1}(t_{k+1}) \quad (22)$$

Based on the definition of Φ in (21), we get:

$$\Phi(t - t_{k+1}) = \prod_{i=1, \dots, k} F_i(t_i)$$

and from (20) and (22), the claim of the theorem follows. $\square$

Based on this result, one can also extend the result from Theorem 2 to discrete distributions which are more likely to be found in a real setting.

VI. RESULTS

We have implemented the algorithm described in Lemmas 1 and 2 for determining upper bounds for the expected completion times for general DAG structures, for both VFI and SSV system. For the case of general component graphs with feedback loops or cycles, we have applied the technique from [6][18] for replacing strongly connected components (SCCs) of the graph with *supernodes*. As shown in [18], the processing rates of these supernodes (and thus, their latencies in cycle counts) can be found by averaging across all nodes in the SCC. The complexity of this conversion algorithm is linear in the number of nodes and edges in the component graph.

Figure 4. Design drivers: (a) Software-defined radio; (b) MPEG-2 encoder; (c) Network-on-Chip based implementation of MPEG-2 encoder; here, in the finest-grain VFI implementation (VFI), each module is in a separate VFI. In the clustered case with two VFIs (VFI-C), modules with the same shade are part of the same VFI.

Based on the upper bound (tree-like) component graph counterparts, we have implemented the algorithms for determining the upper bounds for completion time in *SSV* and *VFI* implementations, as shown in Lemmas 1 and 2, respectively. The

two algorithms require $O(n^2)$ multiplications and $O(n)$ convolutions (only for Lemma 1) and thus have manageable time complexity (n is the number of PEs/cores). While *cdf* multiplication is not time consuming, there are also several efficient implementations for the convolution operation [26][27].

To assess the validity of the presented theoretical results, we have considered two applications that are typically latency constrained: software defined radio and MPEG-2 encoder. As shown in Figure 4., both have series-parallel topologies, while MPEG-2 encoder (Figure 4.(b)) includes a feedback loop. The three nodes involved in the corresponding SCC (*Pred+Add*, *DCT+Quant*, and *IDCT+Iquant*) can be lumped into a single node, such that the resulting topology in this case is a simple series combination.

Both design drivers are characterized by a simple linear pipeline topology, and thus, the *cdfs* as described in Lemma 1 and 2 are exact results, and not bounds. We have also considered the case of a more complex DAG topology, as shown in Figure 4.(c). In this latter case, the MPEG-2 encoder is mapped onto a Network-on-Chip (NoC) platform [28]. The *DCT+Quant* and *IDCT+Iquant* modules are split into two different modules for achieving better performance. For the NoC implementation, we consider two different granularities for the VFI system: VFI - in which each module is in a separate VFI; and VFI-C, in which only two distinct VFIs are used (one comprising of *Source*, *Motion Estimation*, *VLC* and *Sink*, and another comprising of the rest of the modules in the MPEG2-encoder).

Figure 5. From generic DAG to its tree-like counterpart in case of NoC-based MPEG-2 encoder.

In the NoC case, the original topology of the component graph for MPEG2 encoder is shown in Figure 5.(a). As it can be seen, this topology has multiple overlapping paths, and in this case, the bounds from Lemma 1 and 2 are likely to be proper bounds. After applying the approach from Lemma 1 (and Lemma 2), some of the nodes are duplicated (*Source*, *Pred+Add*, *Motion Estimation*) such that no overlapping paths exist. For estimating the communication delays, we used a simple metric that computes the latency using the number of hops and the bandwidth available at each router. This simple metric assumes that there is no congestion in the network. We show in Figure 5.(a)-(b) the equivalent component graph

resulting from our assumption which was used for the analysis. Alternatively, one can use recently published analytical results for communication latency in NoCs [29]. We note, however, that irrespective of the type of NoC used, applying the algorithms to the resulting component graphs would have determined the same qualitative result of VFI being superior to SSV in terms of performance yield, barring any corner cases in which the non-strict inequalities from Lemmas 1 and 2 become equalities.

For each module part of software defined radio or MPEG-2 encoder, we have assumed the same cycle counts and the core information as the ones in [6]. We show in Table I the profiling information for each module part of *software defined radio* and *MPEG-2 encoder*. The nominal clock speed of the Hitachi SH3 cores was 60MHz, while the one for the ARM7TDMI cores it was 133MHz. To model manufacturing process-driven variations, we have assumed that the cycle time is distributed normally, with mean value given by the nominal critical path delay, and standard deviation of 20% (as shown in [30], by 2008 and beyond, gate length variation, and thus delay variation, will reach and possibly exceed this limit). Note that our approach is *general* and can deal with any type of *generic distributions*, not only normal distributions (which have been chosen only for the purpose of illustrating how our approach works).

TABLE I: Driver Application Characteristics

SwRadio Modules (Hitachi SH3 cores)	LPF	Demod	Eq (10)	Sink		
Cycles/packet	61494	33086	463193	32736		
MPEG2-Enc Modules (ARM7TDMI cores)	Motion Est	Pred+Add	DCT+Q	VLC	IDCT+IQ	Sink
Cycles/macroblock	101282	16722	370060	43222	351259	3188

We have computed the *cdf* for each module and overall system level *cdf* for completion time of each application shown in Figure 4.(a)-(b) for both SSV and VFI cases (Figure 7. and Figure 6.). While the latencies of each module in Figure 7.(a) and Figure 6.(b) vary by a lot, the actual per core frequencies (or cycle times) *are the same* in the baseline SSV design. The difference in latencies merely denotes the difference in *cycle counts*, not *cycle times*. As it can be seen in the figures, results verify empirically our proven theoretical result by showing the VFI systems are more likely to satisfy a given latency constraint. Considering the 50% yield target for SSV completion time as the baseline, we note that VFI systems are 80% (software radio) to 90% (MPEG-2 encoder) more likely in satisfying the given latency constraint, achieving a 90-95% yield. In the case of an 80% yield target for SSV completion time, VFI systems have a 99% yield (or 24% more likely in satisfying the same latency target), while for a 90% yield target for SSV completion time, VFI systems ensure a 100% yield (or 11% more likely) in meeting the given latency constraint.

We note that for both systems the computation was exact, since they both exhibit a linear pipeline structure, with no overlapping paths. We have also considered the case of MPEG-2 encoder mapped onto the NoC platform as in Figure 4.(c) where we can expect to see how well the upper bounds computed as in Lemma 1 and Lemma 2 perform. We have compared Monte Carlo simulation for 10M of samples and the bounds provided by Lemmas 1 and 2 for both the SSV and VFI implementations. The results are shown in Figure 8.

Figure 6. Module latency (top) and system level timing yield (bottom) for MPEG2-encoder.

Figure 7. Module latency (top) and system level timing yield (bottom) for software-defined radio.

Figure 8. System level timing yield for MPEG2-encoder mapped onto a Network-on-Chip as in Figure 4.(c).

Figure 9. System level timing yield for MPEG2-encoder mapped onto a Network-on-Chip as in Figure 4.(c) and assuming a discrete set of available clock speeds per system and each VFI, respectively (16, 32, or 64 steps).

As it can be seen, for the VFI case, the upper bound is very tight and almost indistinguishable from the Monte Carlo simulation results. In the case of SSV systems, the upper bound is 3% larger than the nominal exact value obtained via simulation, thus providing a conservative bound for a latency constrained system. For a 50% yield target considering SSV latency as the baseline, the VFI implementation performs 86% (exact MC analysis) or an estimated 90% (bound-based estimation) better, with an overall 93-95% yield. For 80% yield target for SSV, VFI achieves close to 98% likelihood for meeting the same latency (or 23% better yield), while for 90% yield target for SSV, VFI performs 11% better (achieving 100% yield).

Since the theoretical results of this paper also hold for discrete distributions, we also implemented the algorithms for determining the bounds for VFI and SSV systems in a more realistic scenario where clock speeds and voltages take values from a discrete set. In this case, we have considered that clock speeds can be changed between a minimum and maximum value, but only using a

Figure 10. System level timing yield for MPEG2-encoder mapped onto a Network-on-Chip as in Figure 4.(c), assuming a discrete set of 64 available clock speeds per system and varying communication-computation ratios.

Figure 11. System level timing yield for MPEG2-encoder mapped onto a Network-on-Chip as in Figure 4.(c) and assuming a SSV implementation (SSV), a fine-grain VFI implementation with one module per VFI (VFI), and a clustered implementation with two VFIs (VFI-C) as shown by Figure 4.(c).

predefined set of 16, 32, and 64 steps. Assuming this holds for both the whole system (in the case of SSV) and each VFI (in the case of VFI design), we have applied the algorithms from Section V. to the NoC implementation of the MPEG-2 Encoder from Figure 4.(c). We show in Figure 9. the results obtained in this case for the cumulative probability distribution.

As it can be seen, as the number of speed steps increases from 16 to 32 and 64, the expected timing yield increases in both SSV and VFI cases, Furthermore, in the VFI case, since each island can have its speed tuned locally, the differences between steps are smoother than in the case of SSV systems, thus making it closer in behavior to the continuous distribution case. In each of the cases (16, 32, or 64 steps), the experimental results show that the stochastic ordering between SSV and VFI systems still holds: in the case of the 50% timing yield latency value for the SSV baseline, VFI systems have 35-38% better yield than their SSV counterparts. If the 90% timing

yield latency is considered instead, the VFI case recovers the loss in yield and ensures 100% likelihood of meeting a given latency constraint.

The theoretical results presented in this paper hold in the case of negligible communication overhead in the VFI system implementation. In such a case, the predicted improvement in yield can be used to check whether the VFI system can tolerate a given inter-VFI synchronization penalty. However, for a comprehensive treatment, we considered the case of a NoC implementation with a discrete set of 64 clock speed values and three different target yield values for the original SSV implementation: 25%, 50%, and 75%. For each of these cases, we considered the impact of different communication-computation ratios (defined as (#communication cycles / #computation cycles)) on the yield of the final VFI implementation. As it can be seen in Figure 10., increased communication overhead does decrease the timing yield of the VFI system by 6-16%, depending on the baseline yield values (larger decrease for 25% yield values and smaller ones for 75% yield values). However, the resulting timing yield of the VFI system is still larger than the original SSV system: 34-50% for 25% original yield in SSV system; 64-80% for 50%; and 88-94% for 75% original yield in the SSV system. In all cases, the communication/computation ratio varies between zero and 2.5, with larger resulting yield values for the smaller ratios.

Finally, we have used the algorithms specified by Lemma 1 and Lemma 2 in a clustered implementation, where there may be multiple PEs or cores per VFI. We consider again the case of the NoC implementation from Figure 4.(c), but include both the finest-grain VFI implementation with one core per VFI (VFI), as well as a two-VFI implementation where cores are grouped together as shown by the shaded areas of Figure 4.(c) (VFI-C). We show in Figure 11. the system level timing yield for the VFI and VFI-C implementations, along with the SSV implementation. The curves for VFI and SSV have been obtained via Monte Carlo simulation and are the same as the ones in Figure 8., while for the clustered implementation, both our approach (marked by “Bound” in the legend) and the simulation (MC) are included. As it can be seen, a clustered implementation improves yield by 28% when compared to the SSV implementation at the 50% target yield, but it is approximately 8% worse than the finest grain VFI implementation. Furthermore, the bounds are about 5% larger than the Monte Carlo simulation results for the clustered version, showing that indeed the approach is conservative in terms of yield estimation. The results show that our approach can be successfully applied to design exploration for determining the best granularity for a VFI system.

VII. CONCLUSION

This paper has provided a first step in the direction of analyzing statistical properties at *system level* in the presence of variations. To this end, we have described a general approach for characterizing global system level timing yield for complex systems, relying on generic distributions characterizing individual modules. We also showed formally and empirically that VFI-based latency-constrained systems are always more likely to meet given timing constraints than their SSV counterparts, under the same communication latency assumption. Future work will include extensions to address the treatment of rate-based constraint systems and power variability system level analysis.

VIII. REFERENCES

- [1] X. Li, J. Le, P. Gopalakrishnan, L.T. Pileggi, “Asymptotic Probability Extraction for Non-Normal Distributions of Circuit Performance,” in Proc. IEEE/ACM Intl. Conference on Computer Aided Design (ICCAD), San Jose, CA, Nov. 2004.
- [2] M. Orshansky and A. Bandyopadhyay, “Fast Statistical Timing Analysis Handling Arbitrary Delay Correlations,” in Proc. ACM/IEEE Design Automation Conference (DAC), San Diego, CA, June 2004.
- [3] H. Chang, S.S. Sapatnekar, “Statistical Timing Analysis Considering Spatial Correlations Using A Single PERT-like Traversal,” in Proc. IEEE/ACM Intl. Conference on Computer Aided Design (ICCAD), San Jose, CA, Nov. 2003.
- [4] I.A. Ferzli, F.N. Najm, “Statistical Estimation of Leakage-Induced Power Grid Voltage Drop Considering Within-Die Process Variations,” in Proc. ACM/IEEE Design Automation Conference (DAC), Anaheim, CA, June 2003.
- [5] D. Marculescu, 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), San Jose, CA, Nov. 2006.
- [6] K. Niyogi, D. Marculescu, “Speed and Voltage Selection for GALS Systems Based on Voltage/Frequency Islands,” in Proc. ACM/IEEE Asian-South Pacific Design Automation Conference (ASPDAC), Shanghai, China, Jan. 2005.
- [7] H. Wu, I.-M. Liu, M.D.F. Wong, Y. Wang, “Post-Placement Voltage Island Generation Under Performance Requirement,” Proc. IEEE/ACM Intl. Conference on Computer Aided Design (ICCAD), San Jose, CA, Nov. 2005.
- [8] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, “Architecting Voltage Islands in Core-based System-on-a-Chip Designs,” in ACM/IEEE Intl. Symposium on Low Power Electronics and Design (ISLPED), Newport Beach, CA, Aug. 2004.
- [9] A.W. Shogan, “Bounding distributions for a stochastic PERT network,” Networks 7 (1977) 359–381.
- [10] B. Dodin, “Bounding the project completion time distribution in PERT networks,” Oper. Res. 33 (4) (1985) 862–881.
- [11] W. Kleinöder, “Stochastic analysis of parallel programs for hierarchical multiprocessor systems,” Ph.D. Thesis, University of Erlangen, Nürnberg, 1982.
- [12] M. Colajanni, F. Lo Presti, S. Tucci, “A hierarchical approach for bounding the completion time distribution of stochastic task graphs,” Performance Evaluation 41 (2000) 1–22.
- [13] J.Y. Brunel, W.M. Kruijtzter, P. Lieverse, K.A. Vissers, “YAPI: Application modeling for signal processing systems,” in Proc. ACM/IEEE Design Automation Conference (DAC), Los Angeles, CA, June 2000.
- [14] “IBM Blue Logic Cu-08 voltage islands” http://www.ibm.com/chips/products/asics/products/v_island.html.
- [15] L.S. Nielson, C. Niessen, J. Sparso, and K. Van Berkel, “Low-power operation using self timed circuits and adaptive scaling of the supply voltage,” in IEEE Transactions on Very large Scale Integration (VLSI) Systems, 2 (4):391–397, December 1994.
- [16] J. Mutersbach, T. Villiger, and W. Fichtner, “Practical design of globally asynchronous locally synchronous systems,” in Proc. Intl Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), April 2000.
- [17] T. Chelcea and S.M. Nowick, “A low latency fifo for mixed-clock systems,” In Proc. of IEEE Computer Society Workshop on VLSI (WVLSI), April 2000.
- [18] A. Dasdan, “Rate Analysis of Embedded Systems,” Ph.D. thesis, University of Illinois at Urbana Champagne, 1998.
- [19] K.A. Bowman, S.G. Duvall, J.D. Meindl, “Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration,” in IEEE Journal of Solid State Circuits (JSSC), 37 (2):183–190, Feb. 2002.

- [20] E. Humenay, D. Tarjan, K. Skadron, "Impact of Process Variation on Multicore Performance Asymmetry," in Proc. of IEEE/ACM Design, Automation, and Test in Europe Conference (DATE), Nice, France, April 2007.
- [21] S. Malik, M. Martonosi et al., "Static Timing Analysis of Embedded Software," in Proc. ACM/IEEE Design Automation Conference (DAC), Anaheim, CA, June 1997.
- [22] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, C. Spanos, "Modeling within-die spatial correlation effects for process-design co-optimization," in Proc. IEEE Intl. Symposium on Quality of Electronic Design, San Jose, CA, March 2005.
- [23] D. Stoyan, "Comparison Methods for Queues and Other Stochastic Models," John Wiley, New York, 1983.
- [24] R.E. Barlow, F. Proshan, "Statistical Theory of Reliability and Life Testing," Hold, New York, 1975.
- [25] F. Lo Presti, M. Colajanni, S. Tucci, "Stochastic bounds on execution times of parallel computations," in Proc. of the IEEE Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Durham, NC, February 1994.
- [26] A. Devgan, C. Kashyap, "Block Based Statistical Timing Analysis with Uncertainty," in Proc. IEEE/ACM Intl. Conference on Computer Aided-Design (ICCAD), San Jose, CA, Nov. 2003.
- [27] S. Garg, S. Tata, R. Arunachalam, "Static Transition Probability Analysis Under Uncertainty," in Proc. IEEE Intl. Conference on Computer Design (ICCD), San Jose, CA, Oct. 2004.
- [28] J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch (eds.), "Interconnect-Centric Design for Advanced SoCs and NoCs," Kluwer Academic Publishers, April 2004.
- [29] U. Ogras, R. Marculescu, "Analytical Router Modeling for Networks-on-Chip Performance Analysis," in Proc. IEEE/ACM Design, Automation, and Test in Europe Conference (DATE), Nice, France, April 2007.
- [30] S. Nassif, "Design for variability in DSM technologies," in Proc. IEEE International Symposium on Quality Electronic Design (ISQED), March 2000.

and a Best Paper Award from IEEE Asia South-Pacific Design Automation Conference (ASPDAC 2005). Diana Marculescu was an IEEE-Circuits and Systems Society Distinguished Lecturer (2004-2005) and is currently Chair of the ACM Special Interest Group on Design Automation (SIGDA) and member of IEEE. Her research interests include energy aware computing, CAD tools for low power systems and emerging technologies.

Siddharth Garg received a B.Tech. in Electrical Engineering from the Indian Institute of Technology Madras in 2004, and an M.S. also in Electrical Engineering from Stanford University in 2005. He is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at Carnegie-Mellon University. His research interests include energy and variability aware design methodologies.