System-Level Process-Driven Variability Analysis for Single and Multiple Voltage-Frequency Island Systems*

Diana Marculescu, Siddharth Garg
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
Email: {dianam, sgarg1}@ece.cmu.edu

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 11-90% 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.

Categories and Subject Descriptors: I.6 [Simulation and Modeling]: Modeling methodologies; B.8.2 [Performance and reliability]: performance analysis and design aids.
General terms: design, performance.
Keywords: variability, voltage-frequency islands.

1. 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 - WID) or across several dies (die-to-die - D2D). 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 is 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 [5]-[7] 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 [5],[7] for each island have been proposed and shown to increase energy efficiency significantly, under given latency or rate constraints. Although implicitly assumed, 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* [8]-[11] and simple combinations of multiplication and/or convolution of individual component distributions.

1.1. Related Work

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 stochastic bounds [2]. Fast heuristics have been proposed for both statistical timing analysis [3] as well as variability aware leakage power estimation [4]. While this is by no means an exhaustive enumeration of existing work in this area, they all address the variability problem at logic/gate

* This research was supported in part by Semiconductor Research Corporation contract no. 2005-HJ-1314.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICCAD'06, November 5-9, 2006, San Jose, CA
Copyright 2006 ACM 1-59593-389-1/06/0011...$5.00

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.

1.2. Paper Contribution

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 [12], but neither cases considered process variability impact on overall timing yield or latency variability.

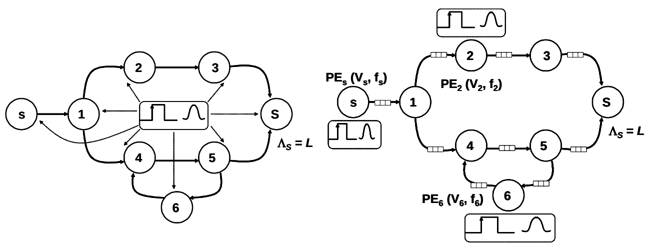
The main contribution of the paper is to provide a first step in the direction of analyzing statistical properties at system level in the presence of variations. Specifically, the approach presented in this paper:

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

2. 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) (b)

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 [5] with cores (PEs) characterized by local speeds/voltages.

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 [13]. 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 [14][15]. 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 [16] 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 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 *process variability* has on performance (and associated energy trade-offs) for applications implemented as systems on a chip, using either a SSV or VFI organization. Since variability affects timing yield, natural *design drivers* for the considered problem are applications characterized by *real time rate* and *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.

3. 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 $\mathcal{V} = \{1, 2, ..., n\}$ 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* [5],[17]. As shown in [17], 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 cycles 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

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

the workload and type of data processed, we restrict ourselves to considering C_i a *constant*[1] characterizing per core *typical* or *worst-case* cycle-count and not a time-varying, workload-dependent variable. This way, we separate the effect of *platform-induced variability* from the effect of *workload-induced variability.* We note that C_i is *application*, and not *architecture* dependent. In other words, it is not related to the number of physical registers/memory elements present in each node of the component graph (in a pipelined or non-pipelined implementation) - the effect of these 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). Indeed, as shown in the FMAX model proposed and validated by Intel [18], the cycle time of as 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, core i latency L_i is also characterized by a *pdf*[2] $f_i(t)$, where $f_i : \mathbb{R} \to [0, \infty)$, and *cumulative distribution function* (*cdf*) given by $Pr(T_i \le t) = F_i(t) = \int_{-\infty}^{t} f_i(u)du$. The *expectation* of each random variable L_i can thus be defined as $E[L_i] = \int_{-\infty}^{\infty} tf_i(t)dt = \int_{0}^{\infty} [1 - F_i(t)]dt$ 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: $\Lambda_i = max_{j \in pred(i)}\{L_j\} + L_i$ where $pred(i)$ 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 $Pr(L \le t) = F(t) = \Phi_S(t)$. 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 *pdf*'s f_i and *cdf*'s 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. However, 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:

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.
2. While the proposed framework assumes continuous distributions throughout the paper, it can be easily extended to discrete distributions.

(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.

3.1. 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 the *cdf* F for the end-to-end latency, 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* [19] and *positively associated variables* [20].

Definition 1. [19] Given two random variables X and Y with positively defined *cdf*'s F_X and F_Y, X is *stochastically smaller* than Y (written as $X \le_{st} Y$ or $F_X \le_{st} F_Y$) if $F_X(u) \ge F_Y(u)$[3] for every $u \ge 0$.

Definition 2. [20] The variables of random vector $X = \{X_1, X_2, \ldots, X_n\}$ are *positively associated* if for all monotone, non-decreasing functions $f, g : \mathbb{R}^n \to \mathbb{R}$ for which expectations exist, $Cov[f(X), g(X)] \ge 0$ where the *covariance matrix* (*Cov*) between random vectors X and Y is defined as $Cov[X, Y] = E[(X - E[X]) \cdot (Y - E[Y])^T]$.

It has been shown [11] that stochastic ordering can be used to characterize upper bounds for positively associated random variables. Indeed, if variables of random vector $\overline{X} = \{\overline{X_1}, \overline{X_2}, \ldots, \overline{X_n}\}$ are an independent version of $X = \{X_1, X_2, \ldots, X_n\}$ (i.e., $\{\overline{X_1}, \overline{X_2}, \ldots, \overline{X_n}\}$ are mutually independent and X_i, $\overline{X_i}$ have the same probability distribution), then $X \le_{st} \overline{X}$, and thus $\overline{X}$ is an upper bound for X.

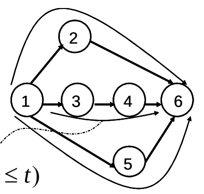
Stochastic ordering is preserved across multiplication and convolution of distributions [19]. If random variables above are path or node latencies in a component graph, note that the *product*

3. Note that we write $F_X \le_{st} F_Y$, even if $F_X(u) \ge F_Y(u)$.

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).

Existing work in determining stochastic bounds for graph completion times [8]-[11],[21] 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 using multiplication and convolution of *cdf's*. 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.

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



$$Pr(L_{1-6} \le t) = Pr(L_{1-5-6} \le t) \cdot Pr(L_{1-2-6} \le t) \cdot Pr(L_{1-3-4-6} \le t)$$

$$Pr(L_{1-3-4-6} \le t) = (F_1 \otimes F_3 \otimes F_4 \otimes F_6)(t)$$

$$Pr(L_{1-5-6} \le t) = (F_1 \otimes F_5 \otimes F_6)(t)$$

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

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* $\overline{\mathcal{G}}$ with paths that *stochastically upper bound* completion times of the original graph $\mathcal{G}$. 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 [21]:

Lemma 1. [11][21][1] Consider a component graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. *Stochastic* upper bounds for node and overall graph completion time can be found from its *associated graph* $\overline{\mathcal{G}}(\overline{\mathcal{V}}, \overline{\mathcal{E}})$ defined as follows:

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

(ii) If *i* is not a source node, let $\overline{\mathcal{G}}_{pred(i)}$ be the graph obtained by considering all $\overline{\mathcal{G}_j}$ in parallel, where *j* is a direct predecessor of *i* in the original component graph. Thus, $\overline{\mathcal{G}_i}$ is defined as the graph connecting $\overline{\mathcal{G}}_{pred(i)}$ in series with node *i*, with *cdf* for node *i* defined as
$$\overline{\Phi}_i(t) = \left[\prod_{j \in pred(i)} \overline{\Phi}_j(t)\right] \otimes F_i(t) \le \Phi_i(t). \qquad (1)$$

(iii) If *j* are all sink nodes in the original graph, $\overline{\mathcal{G}}$ is obtained by considering all $\overline{\mathcal{G}_j}$ 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 $\overline{\mathcal{G}}$ is given by $\overline{\Phi}(t) = \prod_{j \text{sink node}} \overline{\Phi}_j(t) \le F(t)$ which provides a stochastic upper bound for the original graph completion time: $\Lambda_{\overline{\mathcal{G}}} \ge_{st} \Lambda_{\mathcal{G}}$ ❐

1. The proof can be found in [11].

3.2. Completion Time in VFI vs. SSV Systems

The theoretical formulation presented in Section 3.1. provides a means 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*: $L_i / C_i = T$. Thus, the probability that a given path p_j with latency π_j in the component graph satisfies a constraint *t* can be expressed as:

$$Pr(\pi_j \le t) = Pr\left[\bigcap_{i \in p_j,\, \Sigma t_i = t,\, t_i/t_k = C_i/C_k} (L_i \le t_i)\right] \qquad (2)$$

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

$$Pr(\pi_j \le t) = Pr\left[\bigcap_{i \in p_j,\, \Sigma t_i = t} (L_i \le t_i)\right] \qquad (3)$$

While for equation (3) 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 (2). Furthermore, we will show formally that the probability in (2) is smaller than the one in (3), thus making SSV systems less likely to meet timing constraints. 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.[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* $\overline{\mathcal{G}_{SSV}}(\overline{\mathcal{V}}, \overline{\mathcal{E}})$ defined as in Lemma 1, except for equation (1) which is replaced by:

$$\overline{\Phi}_i(t) = \left[\prod_{j \in pred(i)} \overline{\Phi}_j(\alpha_i t)\right] F_i((1-\alpha_i)t) \le \Phi_i(t) \qquad (4)$$

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

We are now able to prove that SSV systems are less likely to meet timing constraints than their VFI counterparts. The result is shown in the following two theorems.

Theorem 1. Consider the distribution (*cdf*) family $(F_1, F_2, \ldots, F_n)$ and associated *pdf*'s $(f_1, f_2, \ldots, f_n)$, $F_i, f_i : \mathrm{R} \to [0, \infty)$. Let $\{\alpha_i\}$ be a set of fixed non-negative numbers with $0 \le \alpha_i \le 1$ and $\sum_i \alpha_i = 1$. Then $\bigotimes_{i=1,\ldots,n} F_i(t) \ge \prod_{i=1,\ldots,n} F_i(\alpha_i \cdot t)$. ❐

2. Remaining proofs can be found in [22].

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 shown in 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:

$$Pr(L_{SSV} \le t) = F_{SSV}(t) \le F_{VFI}(t) = Pr(L_{VFI} \le t) \qquad (5)$$

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

The theoretical results shown in this section can be used to determine how much likelier 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 $L_s \le L_{SSV,y} - L_{VFI,y}$, 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 [5] has shown that this is possible for VFI systems with significant reduction in overall energy requirements.

4. 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 [5][17] for replacing strongly connected components (SCCs) of the graph with *supernodes*. As shown in [17], 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.

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). 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 3., both have series-parallel topologies, while MPEG-2 encoder (Figure 3.(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.

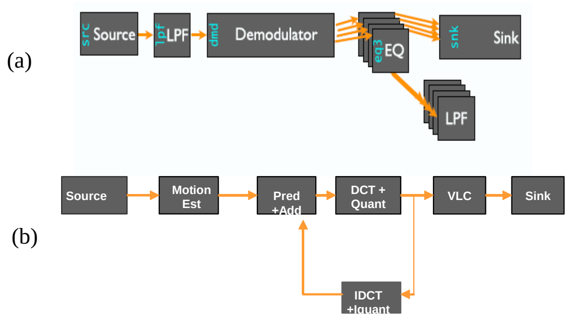


Figure 3. Applications: (a) SW-defined radio; (b) MPEG-2 encoder.

Both design drivers are characterized by a simple linear pipeline topology, and thus, the *cdf'*s as described in Lemma 1 and 2 are exact results, and not bounds. For each module part of software defined radio or MPEG-2 encoder, we have assumed the cycle counts and the core information shown in Table 1 . 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 per domain is distributed normally, with mean value given by the nominal critical path delay, and standard deviation of 20% [23]. However, our approach is *general* and can deal with any type of *generic distributions*, not only normal distributions.

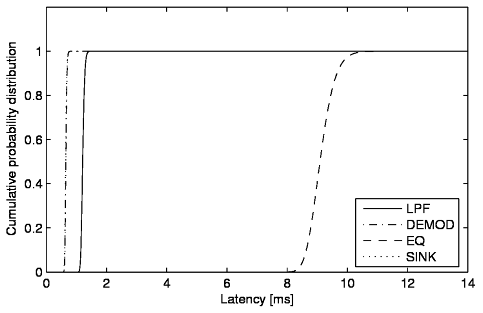
Table 1: Driver Application Characteristics

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 3.(a)-(b) for both SSV and VFI cases (Figure 4. and Figure 5.). 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 than their SSV counterparts. 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.

5. 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.



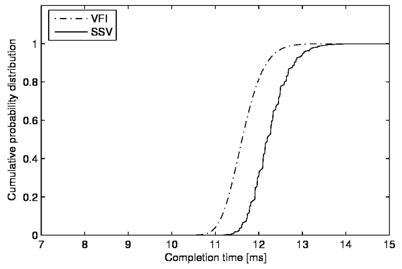
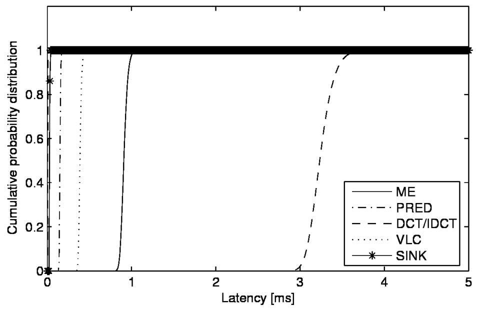


Figure 4. Module latency (top) and system level timing yielc (bottom) for software-defined radio.



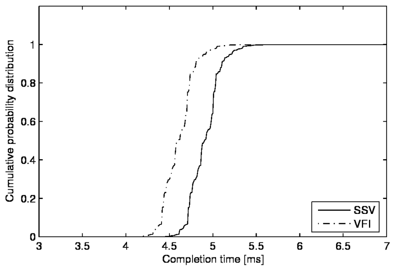


Figure 5. Module latency (top) and system level timing yield (bottom) for MPEG2-encoder.

6. 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] 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.

[6] 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.

[7] 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.

[8] A.W. Shogan, "Bounding distributions for a stochastic PERT network," Networks 7 (1977) 359–381.

[9] B. Dodin, "Bounding the project completion time distribution in PERT networks," Oper. Res. 33 (4) (1985) 862–881.

[10] W. Kleinöder, "Stochastic analysis of parallel programs for hierarchical multiprocessor systems," Ph.D. Thesis, University of Erlangen, Nürnberg, 1982.

[11] 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.

[12] J.Y. Brunel, W.M. Kruijtzer, 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.

[13] "IBM Blue Logic Cu-08 voltage islands" http://www.ibm.com/chips/products/asics/products/v island.html.

[14] 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.

[15] J. Muttersbach, 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.

[16] 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.

[17] A. Dasdan, "Rate Analysis of Embedded Systems," Ph.D. thesis, University of Illinois at Urbana Champagne, 1998.

[18] 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.

[19] D. Stoyan, "Comparison Methods for Queues and Other Stochastic Models," John Wiley, New York, 1983.

[20] R.E. Barlow, F. Proshan, "Statistical Theory of Reliability and Life Testing," Hold, New York, 1975.

[21] 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.

[22] D. Marculescu, S. Garg, "Variability Analysis for Single and Multiple Voltage-Frequency Island Systems," Technical Report CMU CSSI-06-09, April 2006.

[23] S. Nassif, "Design for variability in DSM technologies," in Proc. IEEE International Symposium on Quality Electronic Design (ISQED), March 2000.