Process Variation aware System-level Task Allocation using Stochastic Ordering of Delay Distributions

Love Singhal and Elaheh Bozorgzadeh
University of California, Irvine, California
Email: {lsinghal,eli}@ics.uci.edu

***Abstract*— Design variability due to within-die and die-to-die variations has potential to significantly reduce the maximum operating frequency and effective performance of the system in future process technology generations. When multiple cores in MPSoC have different delay distributions, the problem of assigning tasks to the cores become challenging. This paper targets system level task allocation to stochastically minimize the total execution time of an application on MPSoC under process variation. In this work, we first introduce stochastically optimal task allocation problem. We provide formal theorems of the optimality of the solution in simple scenarios. We extend our theoretical work for generic cases in normal distribution. The proposed techniques enable efficient computation of task allocation using non-stochastic analysis. We apply these techniques in allocating tasks in the embedded system benchmark suites on MPSoC. We show that deterministic solution for system-level task allocation on widely used benchmark topologies and distributions (normal distribution) is almost as good as the best probabilistic solution.**

I. Introduction

The constant reduction in transistor sizes has lead to a state where the transistor sizes (length and width) as well as threshold voltages are not predictable any more. The delays of transistors are now taken as random variables instead of the fixed constant values. The means and variations of these random variables are determined by manufacturing process. Most of the recent work in chip design is now focused on integrating process variation in evaluation of design parameters like critical path delay (SSTA) and leakage power [1]. Recently, researchers have proposed timing yield models and algorithms to incorporate process variations in high level synthesis [2], task scheduling [3] and module selection [4]. A system-level stochastic model has been developed in [5] to incorporate process variability in execution time of the tasks implemented on SoCs with single/multiple voltage frequency islands.

This paper focuses on process variation aware system level task allocation. Task allocation problem assigns each task to a processing core.With high manufacturing variability, new techniques are needed to solve task allocation problems that take into account the variable execution delays of the tasks. Due to the high complexity of variability analysis, it is important to investigate the sensitivity of any design stage to process variability before introducing statistical optimization. There are some design stages such as basic buffer insertion problem, which has shown immunity to process variation [6] and deterministic algorithm performs almost as well as statistical method. In this paper, we focus on system-level task allocation and provide theoretical analysis to show how well deterministic techniques can take the challenge and whether we can tackle the variability of underlying silicon in deterministic fashion.

We first build a theoretical framework to solve task allocation problem in simple scenarios. We provide formal theorems of the optimality of the task allocations on simple topologies when clock periods are stochastically ordered. We extend our theoretical work by focusing on normal distribution, widely used to model the most common stochastic parameters like gate delay. We analyze the distributions to define an ordering between random variables. Our proposed ordering is inspired by stochastic ordering but within an interval. Although our problem formulation uses stochastic variables, the solution can be found in a non-probabilistic manner. The time-consuming computations like convolutions and multiplications of distribution functions of random variables are avoided, thus providing an efficient speed up to the system synthesis tools.

The experimental results on Embedded System Benchmark Suite [7] show that as far as the system level task allocation problem is concerned, the deterministic solutions can work almost as good as the best probabilistic solutions on Gaussian distributions. This is especially true when clock period distributions have the same variance to mean ratio, as widely applied in several proposed process variation aware design tools [2], [5], [8]–[10].

II. Preliminaries

A. Target System Architecture

Modern System-on-Chip devices are comprised of multiple heterogeneous processors together with several hardware IP cores (referred to as MPSoC). The IP core can be a specialized hardware that implements the functionality best suited for the task or can be programmable fabric like FPGA that is configured to implement the task. A set of cores (processors or hardware units) can be connected through a shared bus or shared memory. In this paper, a core of MPSoC is referred to as *Processing Element (PE)*. Each PE has different clock frequency. The frequencies of the PEs are independent of each other. Since clock period at each PE is a random variable due to process variation, execution time of each task on each PE is a random variable. Hence, in order to evaluate the execution time, metrics such as yield and stochastic execution time are deployed. Next, we discuss the delay model and probability distribution function for execution time.

B. System-level Execution Time Model

Our delay distribution model builds upon the generic critical path (GCP) and maximum frequency distribution (FMAX) [1]. It uses principal component analysis to represent the delay of the gate as the sum of the various sources of variations. Assuming that the source of variation is the gate length (other sources can also be included in the model), the delay of the gate is given by the equation,

$$D = D_0 + A_{fit} \times \Delta L_{D2D} + A_{fit} \times \Delta L_{WID} \quad (1)$$

In equation 1, D_0 is the nominal delay, A_{fit} is the delay-length sensitivity, ΔL_{D2D} and ΔL_{WID} are the die-to-die variations and within-die variations. We use the model similar to [5] and [9] to compute the total execution time of the task graphs. We assume that there are multiple cores (PEs) on a chip where each PE is working at its own clock frequency and PEs communicate using special point to point mixed-clock FIFO connections [5]. The objective in this system is to assign tasks to the cores to reduce total system completion time.

III. PE-Task Allocation Problem

We are given a set of tasks or processes of an application to implement on the target system with the objective that the total latency of the system is minimized. The latency of each PE a is characterized by *cumulative distribution function (cdf)* given by : $Pr(T_a \leq t) = F_{T_a}(t) = \int_{-\infty}^{t} f_{T_a}(u)du$. Given the task cycle count c on a PE and delay distribution T of the PE, the total execution

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

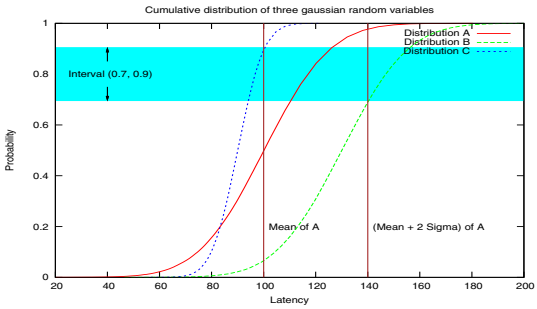

Fig. 1. Stochastic Ordering of 3 Delay Distribution

time of the task on the PE, $L = c \times T$, will also be a random variable, where c is a fixed number. The cycle counts are determined by either the architecture of the PEs or the number of instructions in the software run of the tasks. The task allocation addressed in this paper is a one-to-one mapping which allocates each task to a PE. We assume that the communication delay is same for all task assignments.

The system-level PE-Task allocation problem is defined as:

Given a set of PEs $\{PE_1, PE_2, ..., PE_n\}$ such that their corresponding clock periods $\{T_1, T_2, ..., T_n\}$ are random variables; and given a set of n tasks with their corresponding cycle counts on PEs, allocate each task to a PE such that the total execution time of the task graph on the target system is minimized.

In this paper, we provide a technique for finding task allocation on PEs to minimize total run time when delays are probabilistic. Our intention is to use deterministic (non-probabilistic) technique to assign tasks on PEs. In the next section, we first build a theoretical framework to solve task allocation problem in simplistic scenarios.

During our theoretical analysis, we assume that the cycle count of each task on different PEs is the same. For example, in homogeneous MPSoC that contains replication of same kind of cores, this is held true.

IV. PE-Task Allocation for Stochastically Ordered Delay Distributions

In this section, we assume that clock periods of the PEs are stochastically ordered (s.o.). Stochastic ordering is defined as follows:

Definition 1 (Stochastic Ordering): [5] Given two random variables X and Y with positively defined *cdf*'s F_X and F_Y, X is *stochastically lower* than Y (written as $X \leq_{st} Y$) if $F_X(u) \geq F_Y(u)$ for every $u \geq 0$.

This requires that graphically, the cdf of one distribution is always on the right side of the cdf of other distribution. Figure 1 shows cdfs of three random variables (r.v.), A, B, and C that are stochastically ordered. A stochastically lower random variable has a higher probability of meeting any latency constraint than the stochastically higher random variable. Our objective is to find the task allocation with the *stochastically minimum* total execution time. This inherently means to maximize the likelihood that the system will satisfy the given total execution time constraints. This probability is referred as *yield* at the given latency.

We assume that the delay distribution of any two PEs, say T_a and T_b, are such that they have same cumulative distribution function (cdf) but different means, i.e. $F_{T_a}(t) = F_{T_b}(t + x)$ where x is a constant value. In this case, the means are shifted by x, such that $E(T_b) = E(T_a) + x$. This can happen when multiple instances of a core see their mean clock periods shifted due to WID variation. Distributions A and B in Figure 1 have such relationship. We assume that delay random variables are also independent of each other.

A. PE-Task allocation for Tasks in Series

We first solve the task allocation problem of two tasks and two PEs optimally. Let the latency (total execution time) of the task 1 on PE a be L_{1a}, where L_{1a} is a r.v. Similarly, we define latency r.v. L_{1b}, L_{2a} and L_{2b}, for the two tasks 1 and 2 and the two PEs a and b. The latencies of the random variables [9] are given by equations:

$$L_{1a} = c1 \times T_a; \;\; L_{1b} = c1 \times T_b; \;\; L_{2a} = c2 \times T_a; \;\; L_{2b} = c2 \times T_b \quad (2)$$

In these equations, only T's are the r.v. on the right hand side and c's are fixed predetermined constants. Assuming that the tasks are connected in series, the latency of the overall application is the sum of the latencies of each task on the assigned PEs. Thus, if task 1 is assigned to PE a and task 2 is assigned to PE b, the overall execution time $L_{1a,2b}$ of the system is given by equation: $L_{1a,2b} = L_{1a} + L_{2b}$.

In order to compute the cumulative distribution function (*cdf*) of sum of two r.v., the standard approach is to compute the convolutions of the *cdf*'s of the two r.v. [11].

Definition 2: Given two random variables X and Y with positively defined *cdf*'s F_X and F_Y, the convolution of X and Y is defined as:

$$(F_X \otimes F_Y)(u) = \int_0^u F_X(u') \times F_Y(u - u')du' \quad (3)$$

Thus, if $F_{L_{1a}}(t)$, $F_{L_{2b}}(t)$, and $F_{L_{1a,2b}}(t)$ represent the *cdf*'s of L_{1a}, L_{2b}, and $L_{1a,2b}$, respectively, then:

$$F_{L_{1a,2b}}(t) = Pr(L_{1a,2b} \leq t) = (F_{L_{1a}} \otimes F_{L_{2b}})(t) \quad (4)$$

There are two possible task allocations of two tasks on two PEs - $L_{1a,2b}$ or $L_{1b,2a}$.

Theorem 1 (PE-Task Allocation for 2 Tasks in Series): Given two PEs, a and b, with clock periods T_a and T_b such that T_a and T_b are *stochastically ordered* independent random variables with same distributions and $F_{T_a}(t) = F_{T_b}(t + x)$ where x is a positive constant; and given two tasks, 1 and 2, connected in series in any order, with cycle counts $c1$ and $c2$ such that $c1 > c2$, then the task assignment of task 1 to PE a and task 2 to PE b gives stochastically minimum total execution time.

Next, we give the task allocation technique for the general case of n ($n > 2$) tasks connected in series. We use the following property of the stochastic ordering for our proof. Given two distribution families (F_1, ..., F_n) and (G_1, ..., G_n) such that $F_i \geq_{st} G_i$, for i=1,..., n, the following inequalities are true [12]:

$$\prod_{i=1,...,n} F_i \;\; \geq_{st} \;\; \prod_{i=1,...,n} G_i, \quad (5)$$

$$\bigotimes_{i=1,...,n} F_i \;\; \geq_{st} \;\; \bigotimes_{i=1,...,n} G_i, \quad (6)$$

where $\bigotimes$ is the convolution operator.

For the n tasks and n PEs task allocation problem, the following lemma is used to derive the proof of the optimal solution.

Lemma 1: Given n tasks in series in any order and given n PEs, such that the clock periods of the PEs are stochastically ordered independent random variables with the same distribution; the highest (lowest) cycle count task is always assigned to the PE with the minimum (maximum) expected value of clock period in the stochastically minimum task allocation.

The following theorem gives the optimal task allocation for n tasks.

Theorem 2 (PE-Task Allocation for n Tasks in Series): Given n tasks in series in any order, and given n PEs, such that the clock periods of the PEs are stochastically ordered independent random variables with the same distribution; the task allocation which assigns tasks in the decreasing order of their cycle counts, to the PEs in the increasing order of the expected values of their clock periods gives the stochastically minimum overall execution time.

The next subsection discusses the tasks connected in parallel.

B. PE-Task Allocation for Parallel Tasks

When two tasks are connected in parallel, the combined execution time of the two tasks is the *maximum* of the execution time of each task.

Definition 3: [11] The maximum of two independent r.v. X and Y is a r.v. Z if $F_Z(t) = F_X(t) \times F_Y(t)$ for all values of t.

Interestingly, same technique can be used for finding the PE-task allocation for two tasks connected in parallel as the technique for two tasks connected in series. For this, we assume that the *cdf*'s of delay distributions can be closely approximated to non-decreasing concave functions.

Theorem 3 (PE-Task Allocation for 2 Parallel Tasks): Given two PEs, a and b, with clock periods T_a and T_b such that T_a and T_b are stochastically ordered independent random variables with same distribution and $F_{T_a}(t) = F_{T_b}(t+x)$ where x is a positive constant; and given two parallel tasks, 1 and 2, with cycle counts $c1$ and $c2$ such that $c1 > c2$, then the task assignment of task 1 to PE a and task 2 to PE b gives the stochastically minimum total execution time.

Next, we give the task allocation technique for the general case of n ($n > 2$) tasks connected in parallel. For allocation of n tasks to n PEs, the following lemma and theorem are used to derive the optimal solution.

Lemma 2: Given n parallel tasks and given n PEs, such that the clock periods of the PEs are stochastically ordered independent random variables with the same distribution; the highest (lowest) cycle count task is always assigned to the PE with the minimum (maximum) expected value of clock period in the stochastically minimum task allocation.

Theorem 4 (PE-Task Allocation for N Tasks in Parallel): Given n parallel tasks, and given n PEs, such that the clock periods of the PEs are stochastically ordered independent random variables with the same distribution; the task allocation which assigns tasks in the decreasing order of their cycle counts, to the PEs in the increasing order of the expected values of their clock periods gives the stochastically minimum overall execution time.

Theorems 2 and 4 show that non-stochastic methods (that use only expected values and cycle counts) can be used to achieve optimal task allocation in the stochastic case.

The above discussion pertains to any general distribution of the random variables but under this condition that their corresponding cdfs are only shifted in time. Such special condition may not be applicable in practical cases where clock periods of the PEs do not have same distribution. For example, in normal distribution, two r.v. with different variances do not hold the condition. Next, we relax the condition. In order to provide in-depth analysis, we choose normal distribution and analyze the stochastic ordering under such distribution.

V. PE Task Allocation for Gaussian Distributions

The most common distribution in the real scenarios is Gaussian (or normal) distribution. Gaussian distribution is used to represent the most common stochastic parameters like gate length, width and clock period. The normal distribution and distributions associated with it are analytically very tractable. Under mild conditions, it can be shown that the normal distributions can be used to approximate a large variety of distributions in large samples (using Central Limit Theorem). Researchers have used first order approximations to approximate large number of distributions to normal distribution. Since normal distribution are so widely used, we use special properties of the normal distribution to advance our discussion on ordering of distributions for task allocation. This ordering helps us compute the results efficiently without complex stochastic calculations.

The normal distribution has two parameters, denoted by μ and σ^2, which are its mean and variance. The two parameters provide with complete information about the exact shape and location of the distribution. The probability, $P(|X - \mu| < n\sigma)$, where n is any real number, is same for all normal distributions, irrespective of the values of μ and σ^2, and can be found using common mathematical packages. Before we develop our results for Gaussian distributions, we propose an important extension to stochastic ordering defined in Definition 1.

A. Stochastically Ordered Over an Interval

For strictly increasing cdfs F_X and F_Y of two r.v. X and Y, the stochastic ordering (Definition 1) can also be defined as $X \leq_{st} Y$ if $F_X^{-1}(y) \leq F_Y^{-1}(y)$ for every y in $(0, 1)$. It requires that graphically, the cdf of one distribution is always on the right side of the cdf of other distribution. In real life random variables, this strict condition may not hold true. It is possible that the cdfs of two distributions may intersect each other. Thus, we define a stochastic ordering that exists only over an interval.

Definition 4 (Stochastic Ordering over an Interval): Given two r.v. X and Y with strictly increasing and positively defined cdfs F_X and F_Y, X is *stochastically lower* than Y *over an interval* (u, v), where $0 \leq u \leq v \leq 1$, if $F_X^{-1}(y) \leq F_Y^{-1}(y)$ for y in (u, v).

The *stochastic ordering over an interval* relaxes the constraint of ordering of cdfs (or their inverse function) to an interval. Graphically, this ordering means that the cdf of one r.v. is on the right side of other r.v. over a region parallel to the X axis. Figure 1 shows an example of stochastic ordering of three distributions over the interval $(0.7, 0.9)$. The distributions A and C in the figure are not completely stochastically ordered but are ordered only over an interval. For the computation of yields of different task allocation techniques, we are mainly interested in the interval $(y, 1)$ where y could be 0.9, 0.99 or any region close to 1. We call this interval as *relevant* interval. Any distribution which is stochastically lower than any other distribution in this interval will give the optimum solution at the required yield.

B. Properties of Gaussian Distribution

The cumulative distribution function of Gaussian distribution is a continuous and strictly increasing function. Thus, we can use stochastic ordering over an interval to define the ordering of different Gaussian random variables.

The probability, $P(|X - \mu| < n\sigma)$, where n is any real number, is same for all normal distributions, irrespective of the values of μ and σ^2. For a given y in $(0, 1)$, the value of random variable X (i.e. $F_X^{-1}(y)$) is equal to $\mu_X + n\sigma_X$ for a real number n. The value of n is given by the *quantile* (or *probit*) function of y, which is $n = \Phi^{-1}(y)$, where $\Phi(x)$ is the standard normal cdf. The value of n is only dependent on the value of y. The quantile function is an increasing function and thus as the value of y increases, the value of n increases. For $n = 0$, the value of y is 0.5, for $n = -1$, $y = 0.1587$; and for $n = 3$, the value of y is 0.9987. The point $y = 0.5$ gives mean value and the point $y = 0.9987$ is usually considered as worst-case value.

Thus, for stochastic ordering of two Gaussian r.v. X and Y,

$$\mu_X + n\sigma_X \;\leq\; \mu_Y + n\sigma_Y, \qquad \forall n \in \Re \tag{7}$$

Equation 7 gives us some very important results about the stochastic ordering of two Gaussian r.v. These results are presented in the following cases. Assuming that $\mu_X \leq \mu_Y$, there are following possible cases:

1) $\sigma_X < \sigma_Y$: If for two Gaussian r.v. X and Y, $\mu_X \leq \mu_Y$ and $\sigma_X < \sigma_Y$, then Equation 7 is true for all n in $[\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}, \infty)$. Thus, X is stochastically lower than Y over this interval. The value $\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}$ represents the point of intersection of the two cdfs of X and Y. The two cdfs intersect at point $y = \Phi(\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y})$. Since $\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}$ is a negative number, the yield at the point of intersection is less than 0.5. Hence, for all practical purposes (i.e. for all yields close to 1), the r.v. X is stochastically lower than r.v. Y.

2) $\sigma_X = \sigma_Y$: The case when $\mu_X < \mu_Y$ and $\sigma_X = \sigma_Y$ corresponds to a complete stochastic ordering. The equation 7 is true for all n in $(-\infty, \infty)$. Hence, $X \leq_{st} Y$ when $\mu_X < \mu_Y$ and $\sigma_X = \sigma_Y$. In this case, $F_X(t) = F_Y(t + b)$ where $b = \mu_Y - \mu_X$ for all t. Thus, all the results presented in Section IV are true for this case. It is interesting to see that theoretically this is the only case when any two Gaussian distributions are stochastically ordered (over the whole interval). In all other cases, the point of intersection of

two Gaussian distribution exists, implying that the distributions are not stochastically ordered (s.o.). Thus, two Gaussian distributions are s.o. **if and only if** their variances are same and means are different. However for two Gaussian distributions, the point of intersection could be so close to either $y = 0$ or $y = 1$ that for all practical purposes the two distributions are s.o. even with different variances (especially when there is a large difference in means).

3) $\sigma_X > \sigma_Y$: If for two Gaussian r.v. X and Y, $\mu_X \leq \mu_Y$ and $\sigma_X > \sigma_Y$, then Equation 7 is true for all n in $(-\infty, \frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}]$. Thus, r.v. X is stochastically lower than Y over this interval. And the inverse of Equation 7 is true for all n in $[\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}, \infty)$. Thus, in this case, Y is stochastically lower than X over the interval $[\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}, \infty)$. Since $\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}$ is a positive number, the probability at the point of intersection of two cdfs is greater than 0.5. Now, if the value of $\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}$ is greater than 4 then for all practical purposes, X is stochastically lower than Y.

4) Special Case of σ proportional to μ: A significant number of work considering statistical variation uses the Gaussian distribution for performing experiments. The value of σ is also chosen to be proportional to the value of μ [2], [5], [8]–[10]. In such cases, the delay distributions represent Case (1). A standard technique is to use 20% of μ equal to 3σ. The value of $\frac{\mu_Y - \mu_X}{\sigma_X - \sigma_Y}$ for such a case is equal to -15. The point of intersection of any two distributions will be $y = \Phi(-15) \simeq 0$ on Y axis, which is close to $-\infty$ on X axis. Hence, in such cases, any two random variables are s.o.

Similarly, a recent paper on buffer insertion [6] shows that the σ of the variation in delay per unit length (t_d/l) of wire with buffers remains same around the optimal length of the wire. That is, the value of σ is constant for various μ close to the optimum lengths. Therefore, such situation represents Case (2). Hence, the various delay distributions of the buffered wire are s.o. with same σ and follow results of Section IV [1]. Further, [6] shows that the σ of the delay per unit length (t_d/l) is proportional to the length of the wire when length of the wire is more than the optimal length. Thus, when length of the segment is greater than the optimum length, the delay per unit length (t_d/l) distribution follows Case (4).

Corollary 1: The cdf of any two Gaussian distributions intersect at most at one point (other than positive and negative infinities).

The discussion in this subsection show that it is easy to determine the nature and relationships of various Gaussian distributions by simply analyzing their means and variances. It is easy to determine whether any two Gaussian distributions are s.o. over an interval. Since most designers are only interested in the *relevant* interval (close to 1), they can choose any value y in the *relevant* interval and then compute $F^{-1}(y)$ of the distributions at hand, to determine their ordering. They can also check whether the distributions intersect in the relevant interval to check for any change in the stochastic ordering.

Going back to our discussion on task allocation, we see that we can not use Theorems 2 and 4 for Gaussian delay distributions unless they have same variances. However, we see that many Gaussian distributions for most practical purposes are s.o. over the *relevant* interval. We need to, therefore, find technique to solve the task allocation problem when Gaussian distributions are not completely s.o. The next subsection attempts to solve this problem.

C. PE Task Allocation for Tasks in Series or Parallel

Gaussian distribution has an important sufficient condition for stochastic ordering over the interval $(0.5, 1)$ of two tasks connected in series, which is presented in the following theorem. The following theorem extends Theorem 1 for Gaussian distributions with different values of σ.

[1]Interestingly, their conclusions are also similar to ours. Authors in [6] show that using the mean values of parameters to solve the traditional buffer insertion problem gives same results as using the probabilistic method of solving buffer insertion problem. Our conclusion is similar for task allocation problem. They showed it using empirical results and did not use stochastic ordering.

Theorem 5 (PE-Task Allocation for 2 Tasks in Series): Given two PEs, a and b, with clock periods T_a and T_b such that the T_a and T_b are two independent Guassian random variables and $\mu_{T_a} < \mu_{T_b}$ and $\sigma_{T_a} \leq \sigma_{T_b}$; and given two tasks, 1 and 2, connected in series in any order, with cycle counts $c1$ and $c2$ such that $c1 > c2$, then the task assignment of task 1 to PE a and task 2 to PE b gives stochastically minimum total execution time *over the interval* $(0.5, 1)$.

The proof of this theorem is omitted for sake of brevity. As already discussed, there are two possible task allocations of two tasks and two PEs. The two task allocations will have different distributions which are also Gaussian. Theorem 5 says that if the Case 1 is true for the distributions of two PEs, then the Case 1 is also true for the two distributions of two task allocations. The optimum distribution can be easily found by ordering the values of means, or $F_X^{-1}(y)$ and $F_Y^{-1}(y)$ where y is in the *relevant* interval, of the clock periods. In our work, we therefore propose to use these values in the *relevant* interval, in order to determine the stochastic ordering of the clock periods of the PEs and use this ordering to determine the task allocation. In our experiments, we then compare our results with the optimum solutions to determine the effectiveness of this approach.

For tasks connected in parallel, we could not find any result similar to Theorem 5 for Gaussian delay distribution. One of the reason is that the maximum of two Gaussian distribution is not Gaussian any more. It can be only approximated to a Gaussian distribution (using tightness probability and moment matching techniques [8]), and the resultant approximation is complex. However, in our experiments we see that the same task allocation as Theorem 5 gives stochastically minimum total execution time over the relevant interval even when tasks are connected in parallel. Hence, we use the same task allocation for parallel tasks as well.

VI. Evaluation Results

In this section, we show the comparison between deterministic and probabilistic solutions of task allocation. For deterministic solutions, we evaluate the effectiveness of using mean and worst case values of clock periods.

We implement our platform in Java and experiments were conducted using various benchmarks, including MPEG2 encoder [5] and five embedded synthesis benchmarks with multiple tasks graphs from the E3S suites [7] (from the Embedded Microprocessor Benchmark Consortium (EEMBC)). We use many cores like AMD, IBM Power PC, and Motorola, with mean clock frequencies range from 133 MHz to 500 MHz. We assume Gaussian distributions of clock periods of the cores. Three sets of experiments are performed to show effectiveness of the various results proposed in this paper. All the final execution times are computed using exhaustive simulations that consist of 10000 runs of Monte Carlo simulation. The yield point is chosen to be 99%.

To compare results of deterministic and probabilistic solutions, we find the optimal solution of task allocation by considering all possible task allocations. Three kinds of optimal solutions are computed - *exact*, *at-yield* and *at-mean*. The *exact* optimal solution is computed by considering all possible task allocations, performing Monte Carlo simulations on each of them, and then finding a task allocation which has the lowest total execution time at the yield point. We use two values of y for $F^{-1}(y)$ for all processors, $y = 0.99$ (close to worst-case delay) for *at-yield* and $y = 0.5$ (mean delay) for *at-mean*. The optimal solution that gives minimum total runtime is then found exhaustively. If *at-yield* (or *at-mean*) and *exact* optimal solutions are close, it shows that deterministic solution could achieve as good results as more expensive probabilistic solution.

The first set of results is related to yield improvement on the tasks connected in series. We use all the linear task graphs in the E3S benchmarks and MPEG2 Encoder application for this set. The cycle counts of MPEG2 Encoder are derived from [5]. In this set

TABLE I

EXECUTION TIMES (IN μ SEC) AT THE 99% YIELD POINT OF VARIOUS TASK ALLOCATIONS FOR SERIES TASK GRAPHS FOR STOCHASTICALLY ORDERED DISTRIBUTIONS

Benchmark	*Exact* Optimal	*At-Yield* Optimal	*At-Mean* Optimal
E3S - Telecom0	1414	1414	1414
E3S - Network1	1123	1123	1123
E3S - Network2	1566	1566	1566
E3S - Network3	2531	2531	2531
E3S - Auto1	959	959	959
E3S - Auto3	557	557	557
MPEG2	1249	1249	1249
Average	1342.7	1342.7	1342.7

TABLE II

EXECUTION TIMES (IN μ SEC) AT THE 99% YIELD POINT OF VARIOUS TASK ALLOCATIONS IN REAL SCENARIO

Benchmark	*Exact* Optimal	*At-Yield* Optimal	*At-Mean* Optimal
E3S - Telecom0	122	122	123
E3S - Network1	885	885	885
E3S - Network2	1142	1142	1142
E3S - Network3	1352	1352	1352
E3S - Auto1	96	96	96
E3S - Auto3	78	78	78
MPEG2	1124	1124	1153
E3S - Telecom1	686	686	687
E3S - Auto2	X	2449	2906
E3S - Consumer0	35050	35050	35280
E3S - Automation0	6182	6182	6787
Average	4469.6	4469.6	4589.9

of experiments, we assume that the cycle counts of tasks are same for each PE. We extract the cycle counts of E3S benchmarks by finding the maximum cycle counts of each task among all the used cores. We assume σ to be same for each core where 3σ is 30% of the average clock period of cores. This set of experiments represents application of Theorem 2 and Case (2) of Section V-B.

Table I shows the result of the first set of experiments. All the applications in this table consist of tasks connected in series. Also, the delay distributions of any two PEs are s.o., i.e., they have same variances and different means. This set of experiments is an example of scenario presented in Section IV. Both deterministic solutions *at-yield* and *at-mean* give best probabilistic solutions (*exact* optimal). In fact, use of any y in $F^{-1}(y)$ for computing deterministic optimal solution will give probabilistic optimal solution as distributions are s.o. over whole interval.

Table II shows the second set of results. The second set of experiments represents the most real scenario. We select five different processing cores for the implementation - three AMD cores, one IBM PowerPC core, and one NEC core. We then extract the cycle counts of all tasks on these processing cores. The variance of delay distribution for different cores is chosen from 5% to 30% of the clock periods of the cores. We use both series-only applications and series-parallel graph applications of E3S and MPEG2 benchmarks. The series-parallel graph is a graph which is composed of series or parallel combinations of the smaller series-parallel graphs. The two categories of applications (series and series-parallel) are separated by a horizontal line.

Table II shows that *at-mean* solution is not as effective as *at-yield*. This is because the stochastic ordering of clock periods around point $y = 0.5$ (mean value) is not always same as the stochastic ordering around the point $y = 0.99$ (our yield point). The Gaussian distributions that intersect in interval $(0.5, 0.99)$ will add errors to the optimality of *at-mean* approach. We therefore believe that worst-case values are better than mean values for deterministic analysis in task allocation. This is because in the interval close to the worst-case yield point (99%), the slope of all Gaussian distributions is flat (See Figure 1). On the other hand, the Gaussian distributions around mean values (50% yield) have steep slopes. Thus, the worst-case delays give more reliable solution for task allocation than mean delays.

Another important observation is the accuracy of *at-yield* approach. We see that *at-yield* approach gives the best probabilistic solutions for all benchmarks for this set of experiments as well. The fact that the delay distributions are not s.o. even over the interval $(0.5, 1)$ and the tasks have different cycles for each PE, makes this observation even more interesting. This shows that the optimal deterministic solution of task allocation problem gives results as good as any probabilistic solution. Although these results will depend on the values of mean and variance chosen by us, even for different values of mean and variance, we could not find a case when *at-yield* and *exact* approaches do not give identical results. This is partly due to non-complex topologies of typical embedded system benchmark suite. We therefore believe that as far as the system level task allocation problem targeting typical embedded system application is concerned, probabilistic approaches do not offer any significant advantage over deterministic approaches, when delay distributions are Gaussian.

VII. CONCLUSIONS

In this paper, we propose system-level task allocation techniques on MPSoC under process variation. We use stochastic ordering of delay distributions to determine task allocations effectively. We first provide stochastically optimal task allocation for tasks connected in series or parallel with the formal proofs. We then extend the proposed optimal solution for Gaussian distribution. Our solution uses non-stastistical analysis, and hence, provides fast computation in the system level synthesis stage. We show that, for system-level task allocation problem targeting typical embedded system benchmarks, deterministic solutions can work as well as probabilistic solutions for Gaussian distributions.

REFERENCES

[1] K. A. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Journal of Solid State Circuit*, vol. 37, no. 2, Feb. 2002.

[2] J. Jung and T. Kim, "Timing variation-aware high-level synthesis," in *IEEE ICCAD*, 2007, pp. 317–321.

[3] F. Weng, C. A. Nicopoulos, X. Wu, Y. Xie, and V. Narayanan, "Variation-aware task allocation and scheduling for mpsoc," in *IEEE ICCAD*, 2007.

[4] F. Wang, X. Wu, and Y. Xie, "Variability-driven module selection with joint design time optimization and post-silicon tuning," in *IEEE ASPDAC*, 2008.

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

[6] L. Deng and M. Wong, "Buffer insertion under process variations for delay minimization," in *IEEE ICCAD*, 2005.

[7] R. P. Dick, "Embedded systems synthesis benchmarks suite (e3s)." [Online]. Available: http://www.ece.northwestern.edu/~dickrp/e3s

[8] C. Vishweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *IEEE Design Automation Conference*, 2004.

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

[10] Y. Lin, M. Hutton, and L. He, "Placement and timing for fpgas considering variations," in *IEEE FPL*, 2006.

[11] A. Srivastava, D. Sylvester, and D. Blaauw, *Statistical Analysis and Optimization for VLSI: Timing and Power*. NY, NY: Springer, 2005.

[12] M. Colajanni, F. L. Presti, and S. Tucci, "A hierarchical approach for bounding the completion time distribution of stochastic task graphs," *Performance Evaluation*, vol. 41, no. 1, 2000.