Robustness Analysis of Multiprocessor Schedules

Shreya Adyanthaya*, Zhihui Zhang*, Marc Geilen*, Jeroen Voeten†*, Twan Basten*† and Ramon Schiffelers‡*
*Eindhoven University of Technology, Eindhoven, The Netherlands
†TNO-ESI, Eindhoven, The Netherlands
‡ASML, Veldhoven, The Netherlands
Email: s.adyanthaya@tue.nl, zhihui.zhang@js3.nl, {m.c.w.geilen, j.p.m.voeten, a.a.basten}@tue.nl, ramon.schiffelers@asml.com

***Abstract*—Tasks executing on general purpose multiprocessor platforms exhibit variations in their execution times. As such, there is a need to explicitly consider robustness, i.e., tolerance to these fluctuations. This work aims to quantify the robustness of schedules of directed acyclic graphs (DAGs) on multiprocessors by defining probabilistic robustness metrics and to present a new approach to perform robustness analysis to obtain these metrics. Stochastic execution times of tasks are used to compute completion time distributions which are then used to compute the metrics. To overcome the difficulties involved with the max operation on distributions, a new curve fitting approach is presented using which we can derive a distribution from a combination of analytical and limited simulation based results. The approach has been validated on schedules of time-critical applications in ASML wafer scanners.**

I. Introduction

High performance special purpose platforms have architectures that are designed to be specific to the applications running on them and have the advantage of high predictability. As such, there is little variation in the execution times of the applications running on them. Due to the high cost of making such application specific platforms, current trends show a shift towards general purpose platforms. With the advent of multicores, demands of complex embedded applications can be met also by general purpose platforms. However, these platforms suffer from low predictability and exhibit fluctuations in execution timings. Hence, there is a need to cope with these fluctuations and to be robust in nature.

Scheduling and analysis of applications in classical real time approaches mostly take the worst case execution timings into account. If a static order multiprocessor schedule of an application meets its latency requirements in the worst case, it is highly robust. However in the applications of modern high tech equipment such as wafer scanners, printers or healthcare equipment, execution timings vary in such a way that most occurring (nominal) execution times typically show a big difference with the worst case and are closer to the best case execution times. Scheduling of applications for the worst case is either always infeasible or requires excessive resources to meet latency requirements. Hence scheduling is usually done for the nominal case. However, tasks can violate their latency requirements due to execution time variations. This raises the concern that schedules running on general purpose multiprocessor platforms must be robust to be able to cope with these fluctuations with low probability of resulting in failures.

To produce schedules that are maximally robust against execution time fluctuations, we need to design robust schedulers. This requires two steps. The first step is the analysis of the robustness of schedules. The second step is robust scheduling which uses the robustness analysis to steer scheduling decisions. This work focuses on the first step by establishing an approach that can be used to measure the robustness of schedules of DAGs on multiprocessors and their constituent tasks. Robustness of a task is defined as the probability of meeting its deadline. Robustness of a schedule is then defined as the (normalized) expected value of the number of tasks missing their deadline.

Since nominal execution times of tasks are mostly closer to the best case execution time than the worst case execution time, the probability density function of the task execution time is mostly not normally distributed but right skewed in nature. Multiprocessor schedules are characterized by dependencies that span along as well as across processors. Propagation of completion times along these dependencies requires a combination of convolutions and maximization operations to be performed on the task execution and completion time distributions. Since there are no known practical analytical means to compute the distribution of the maximum of stochastic variables that are skewed in nature (e.g. skew-normal or PERT), we device an approach that combines the advantages of both analytical computations and simulations to accurately estimate the robustness of tasks and schedules. Apart from being skewed, we have information on the bounds of the distributions in the form of best-case and worst-case task execution times. The combination of the skewness and the boundedness requirements is met by the PERT distributions, as opposed to the skew-normal distributions which do not meet the boundedness requirement. We present a new approach that fits a PERT distribution on simulated histograms using analytically computed bounds. The techniques have been tested for scalability on very large applications of ASML wafer scanners.

The paper is organized as follows. Section II gives the related work in robustness analysis. Section III presents some preliminary definitions. Section IV presents the problem definition and summarizes the solutions approach applied in this work. Section V gives the major challenges of the analysis and Section VI gives the details of the proposed practical realization of the analysis. Section VII gives the experimental results and Section VIII concludes the paper.

II. Related Work

A recent paper by Canon and Jeanot [9] evaluates several robustness metrics for DAG schedules and compares them. Among others it discusses a distance metric which measures robustness by comparing the cumulative distribution function

of a certain performance metric with and without perturbations to derive robustness measures [11]. However, in our work, we have a fixed schedule and fixed execution time distributions of tasks and we measure the robustness of the schedule under these given variations. We do not study any changing parameters that can cause perturbations, which makes the distance metric not applicable here. In [3], a metric called the 'robustness radius' is presented that is the smallest variation of a certain parameter that affects the system performance. Such kind of definition does not take into account probabilities that some parameter changes occur more often than others. Another metric is the slack mean which is the mean of the slack in the schedule [8]. Slack-based metrics to quantify robustness are also presented in [17]. These metrics are suitable when we do not have the stochastic execution time information that forms the basis of this paper. In our work, we utilize this stochastic information to derive deadline miss probability-based metrics to quantify DAG schedule robustness.

There is also literature that does express robustness in probabilistic terms. In [22], the deadline miss probabilities of tasks running on unreliable hosts is estimated. This work is different from ours since they study the availability periods of hosts and their effect on tasks meeting their deadlines. In our work we study robustness of schedules in relation to task execution time variations that are known beforehand and internal to the tasks. Another probabilistic metric is presented in [16] that gives the probability that the makespan is within two bounds. This can be used to address the deadline problem by assuming the upper bound on the makespan to be the deadline that needs to be met. However, the approach only looks at independent applications and hence the propagation of distributions avoids the need to consider synchronization and hence the need to compute the maximum of stochastic variables. In our DAG schedules, tasks on different processors have data dependencies between them. When a task has dependencies from two source tasks on different processors, the start time of the tasks is the maximum of the completion times of the source tasks requiring the analysis of the maximum of stochastic completion times.

The central limit theorem [14] states the convergence of many random variables (under certain fairly common conditions) to approximately a normal distribution under summation, but it does not apply to the maximum operator. In fact, it has been observed that the max of standard normal distributions can tend to be skewed depending on the difference in the standard deviations of the input terms. There are no known classes of continuous distributions that are closed under the maximum operation (such that the distribution of the maximum is also in the class). Work has been done to approximate the maximum of 2 or more standard normal random variables with another standard normal random variable up to a certain degree of accuracy such as presented in the work of Clark [10] and in [13], [5]. Computing the exact distribution of the maximum is especially hard in case of correlations among the input distributions. This comes with the cost of increased complexity as presented in [7]. However, we could not find methods of approximating the maximum of distributions that are skewed by nature. In the applications considered in the domain of this paper, the distributions of the execution times of tasks are most often skewed (right skewed with nominal values closer to min). Hence, we cannot use normal approximations without losing accuracy in the results.

Alternatively, one could approximate distributions with a limited number of discrete values. However, this results in exponential computational complexity as all combinations for different tasks need to be enumerated [6]. Another approach to this computation is the enumeration of all the critical paths leading to a particular node. The propagation of distributions along these individual paths follows the sum operator alone. Finally, maximization is applied to the distributions of all the critical paths. This approach separates the analysis into two parts: computing the timings of paths which can be computed with simple convolutions and which enjoy the central limit theorem and finally applying the maximization [2]. The drawback of this approach is the need to find the critical paths to be considered per node. For large schedules the number of critical paths that need to be detected can be very large.

Most approximation methods compare with extensive simulations to judge the accuracy of their approximations [19][15]. This involves simulating a large number of samples to obtain the entire resultant distribution including the tails with often small probability mass that determine the deadline miss probabilities. This has the drawback of being too time consuming. In this work, we also use simulations, but we use the PERT distribution to shorten simulation times. The results of these quick simulations are combined with analytically computed bounds to obtain the distributions.

On the PERT fitting aspect, there exist distance metrics in the literature that can be used to evaluate the goodness of fit of distributions on samples of data [11][12]. However, most of these metrics have a distribution of their own and studying these distributions gives an estimate on the closeness of the input distributions. In our work, we propose a new metric based on the inner product of distributions that returns a single value which gives an estimate of the closeness of two distributions. Combined with a divide and conquer search algorithm, it allows us to accurately and with low computational complexity fit a PERT distribution on the histograms obtained from limited simulations.

III. Preliminaries

A *task* $A \in T \subseteq (\mathbb{R} \rightarrow \mathbb{R}^{\geq 0}) \times R \times \mathbb{R}^{\geq 0}$ is defined by a tuple $A = (p^e_A, r_A, d_A) \in (\mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^{\geq 0}) \times R \times \mathbb{R}^{\geq 0}$ where T denotes the set of tasks, $r_A \in R$ denotes the resource in R, the set of resources, p^e_A denotes the probability density function for the continuous distribution of the execution time of the task A, and d_A denotes the deadline of A. The random variable for the execution time of the task is denoted by e_A.

A *dependency* $(A, B) \in D \subseteq (T \times T)$ between two tasks A and B states that B can begin its execution only after the completion of A.

A *static-order schedule* S is a mapping $S : R \rightarrow T^*$ from the set R of resources to an ordered list of tasks from T. We assume a schedule to be a *fixed binding and execution order of tasks on processors*, according to which we perform robustness analysis of run-time scheduling. Given a schedule, the tasks have (stochastic) start and completion times. For task A, we let s_A and c_A denote the random variables for the start and completion time, respectively. The start time of the first task scheduled on a resource, without any predecessors, is 0. The completion times are derived by adding the task execution

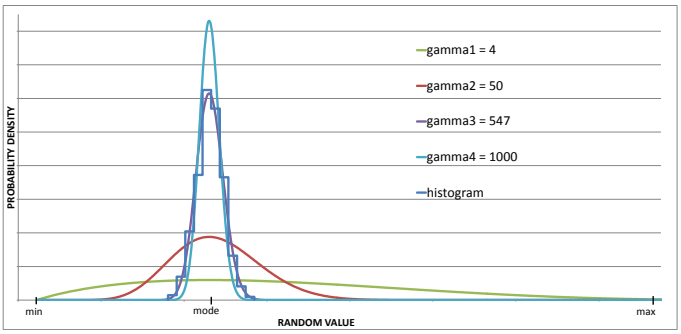

Fig. 1: Modified PERT distributions with different γ values.

times to the corresponding task start times. The probability density functions for the start time and the completion time of A are denoted by p_A^s and p_A^c respectively.

A *PERT distribution* [21] is a version of the Beta distribution and is defined by three parameters, namely the minimum (min), the most likely value ($mode$) and the maximum (max). It derives its name from the PERT (project evaluation and review technique) networks, a statistical tool used in project management to analyse, with respect to their timings, the tasks involved in completing a project.

A *modified PERT distribution* is a variant of the PERT distribution developed by David Vose and allows producing shapes with varying degrees of uncertainty by means of a third parameter, gamma (γ), that scales the width (variance) of the distribution. In the standard PERT, $\gamma = 4$ and upon increasing the value of γ, the distribution becomes more concentrated around the mode. On the other hand on decreasing the value of γ, the distribution gets more spread out between the min and max. Figure 1 shows the standard PERT distribution with $\gamma = 4$ and the modified PERT distributions obtained by increasing the values of γ and their fitting on a histogram for a particular task in the ASML schedules. The probability density function for the modified PERT distribution with min, max, $mode$ and γ as parameters is given by the following equation:

$$p(x) = \begin{cases} \frac{(x-min)^{\alpha_1-1}(max-x)^{\alpha_2-1}}{\beta(\alpha_1,\alpha_2)(max-min)^{\alpha_1+\alpha_2-1}} & min \leq x \leq max \\ 0 & \text{otherwise} \end{cases}$$

where the shape parameters α_1 and α_2 are given in Equation 1 and the beta function ($\beta(\alpha_1, \alpha_2)$) is given in Equation 2.

$$\alpha_1 = 1 + \gamma\left(\frac{mode - min}{max - min}\right); \alpha_2 = 1 + \gamma\left(\frac{max - mode}{max - min}\right) \quad (1)$$

$$\beta(\alpha_1, \alpha_2) = \int_0^1 t^{\alpha_1-1}(1-t)^{\alpha_2-1}dt \quad (2)$$

IV. Problem Definition and Solution Approach

A DAG schedule is a fixed static order and binding of tasks on a multiprocessor. Tasks in a schedule have dependencies between them both along and across processors. We make two additional assumptions: (1) *Task execution time distributions are known and have finite support, i.e., lower and upper bounds outside of which the probability density is zero*. A deadline is given for each task. (2) *Execution times of tasks are independent*. In reality positive or negative correlations between execution times of tasks may exist, but the focus of this work is to study how robustness of schedules is influenced by scheduled order and synchronization of the tasks.

A. Problem Statement

Robustness is a measure of the tolerance of a task or schedule to variations in the execution times of tasks. Tolerance is measured by the probability that the tasks in the schedule still meet their deadlines in the presence of these variations. First, robustness of tasks to missing their deadlines is defined and from that robustness of the schedule as a whole is defined. The statement of the problem being dealt with in this paper is:

"Given static-order schedule S, what is the robustness of S and its constituent tasks?"

B. Challenges

To obtain deadline miss probabilities, we need to derive task completion time distributions from task execution time distributions and the given schedule. There are two obvious approaches to doing this: (1) compute the completion time distributions analytically, (2) use the execution time distributions to draw execution time samples and perform extensive simulations to obtain the full completion time distributions. However, computing the completion time distributions analytically is highly complex due to the presence of the max operations owing to the dependencies between the tasks. This will be further elaborated in Section V. On the other hand performing simulations alone will require us to perform extensive, time consuming simulations to be able to derive full completion time distributions with sufficiently accurate tails to estimate deadline miss probabilities. Instead, we use an approach which combines limited simulations with analytically computed bounds to estimate the completion time distributions.

C. Solution approach: Overview

In this subsection we summarize the overall approach of the paper. Deadline miss probabilities can be derived from the distributions of the completion times of the tasks and their deadlines. Before we can compute the completion time distributions, we need task execution time distributions. These are typically approximated using statistical data from measurements. We present a curve fitting approach that can be used to fit a PERT distribution on histograms obtained from such measurements, as shown in Figure 2(a). Due to the difficulties of performing max and plus operations on distributions [6], this cannot be done entirely analytically. On the other hand, performing only simulations produces insufficient rare deadline misses depending on the length of the simulations as shown in Figure 2(b) and (c). Hence, we instead approximate the completion time distributions as PERT distributions using analytically computed upper and lower bounds and data from limited simulations, carried out by drawing samples from the given PERT execution time distributions. This is shown in Figure 2(d). Once we have obtained the estimates of the completion time distributions, we can calculate task robustness as the red shaded portion of the area under the density function

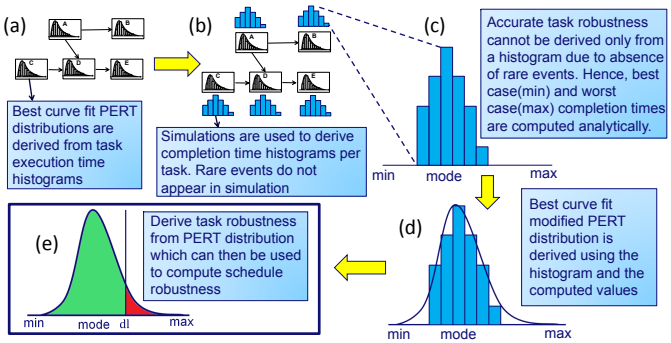

Fig. 2: Robustness analysis: flow of solution approach.

in Figure 2(e). Schedule robustness is then quantified as the expected number of task deadline misses. Although not a focus of this work, it might be possible to adapt the fitting algorithms to apply the overall approach even when the distributions are not PERT-like.

V. Challenges of the Analysis

In this section we explain in some detail the analytical model and the reasons for its complexity. This is followed by an explanation on why an approach using only simulations is also not practically feasible.

A. Analytical approach only

Since we want to compute the deadline miss probabilities of all tasks in the schedule, we need to compute and propagate the completion time distributions per task. Computing these distributions under the maximum operation is very difficult as is explained below. Given the start and execution time distribution of a task A, the completion time distribution is the distribution of their sum and is computed as follows.

$$p^c_A(t) = \int_0^\infty p^s_A(t') \cdot p^e_A(t - t')dt' \tag{3}$$

The start time of a task without predecessors, when there is no task scheduled on its resource r is 0, and its distribution is as follows ($\delta(t)$ represents the Dirac δ-function).

$$p^s_A(t) = \delta(t) \tag{4}$$

The start time of a task which has no predecessors but has tasks scheduled before it on its resource r, with $before(r, A)$ being the last task scheduled on r before A, is the completion time of $before(r, A)$.

$$p^s_A(t) = p^c_{before(r,A)}(t) \tag{5}$$

The start time of a task A with predecessors is the maximum of the completion time of the last completing predecessor ($lastPred(A)$) and that of $before(r, A)$.

$$s_A = \max(c_{before(r,A)}, c_{lastPred(A)}) \tag{6}$$

The corresponding start time distribution is computed as follows.

$$\begin{aligned} p^s_A(t) = p^c_{before(r,A)}(t) \int_t^\infty p^c_{lastPred(A)}(t')dt' \\ + p^c_{lastPred(A)}(t) \int_t^\infty P^c_{before(r)}(t')dt' \end{aligned} \tag{7}$$

If the completion times of $lastPred(A)$ and $before(r, A)$ are dependent, due to the existence of dependencies to common tasks, one would have to resort to computing it from the joint distribution of $lastPred(A)$ and $before(r, A)$.

$$\begin{aligned} p^s_A(t) = \int_t^\infty p^c_{before(r,A),lastPred(A)}(t, t')dt' \\ + \int_t^\infty p^c_{lastPred(A),before(r,A)}(t', t)dt' \end{aligned} \tag{8}$$

Such a joint distribution also includes the information about the correlation between the individual completion time values of the tasks. As such, obtaining this distribution is difficult in practice. The completion time distribution can be computed from the start time distributions using Equation 3. However, obtaining the start time distributions is hard even without correlations. This is because there are no continuous distributions that are known to be closed under the max operation, which captures synchronization on input dependencies and execution times. Hence, even if $p^c_{before(r,A)}$ and $p^c_{lastPred(A)}$ are known distributions, the distribution of their max need not be the same or even a known distribution. Also, if $p^c_{before(r,A)}$ and $p^c_{lastPred(A)}$ are the max of certain other distributions (from earlier in the schedule) then it is possible that their properties are already not known. Due to this it becomes very hard to compute their integrals. Alternatively we could consider discrete enumerations of $p^c_{before(r,A)}$ and $p^c_{lastPred(A)}$ for the computation. We would then need to compute the max (or sum) for each combination of discrete values from them. The complexity of this computation grows exponentially in the size of the task graph and the number of possible discrete values [18]. This is clear when we consider a schedule with n tasks and each task has m possible values for its execution time. The number of possible values for the completion time is m^n. Industrial schedules may have thousands of tasks executing on general purpose platforms and exhibiting large variations in their execution timings. To avoid the exponential complexity of such a discrete approximation, we should instead be able to somehow approximate the resultant distribution to some known distribution with parameters computed in terms of the parameters of the input distributions. However, as already seen in Section II, there are also no known analytical approximations for the parameters of the max of skewed distributions. As such, we could not compute the completion time distributions of tasks analytically alone.

B. Simulations only

Simulations are performed on the schedules by drawing samples from the task execution time distributions. Extensive simulations to obtain a large number of completion time

samples can be used to estimate the entire completion time distributions. The advantage of this approach is that we do not need to keep track of the correlations between the various distributions since they are inherently carried across in the simulations. The main drawback is that extensive simulations require a significant amount of time. In particular, simulating fewer samples results in the drawback that events with very low probability of occurrence (such as deadline misses often are) may not appear at all or too infrequently to accurately estimate their likelihood. As a result, with limited simulations we only obtain values around the most likely completion times and miss out those that are less likely. In this scenario, we will find the probability of missing deadlines to be estimated very inaccurately. Hence, we need an approach that combines the accuracy of the analytical approach to obtain rare events with the strength of simulations to generate mass around the most likely events and to naturally handle correlations, to obtain completion time distributions.

VI. Proposed robustness analysis approach

We assume that task execution time distributions are known apriori. Mostly in reality, the information known about task execution times is limited to measurements. These measured discrete values can be classified into histograms. In order to obtain distributions for task execution times, the histograms are fitted with continuous PERT distributions. The following section explains our approach to fit a PERT distribution on execution or completion time histograms.

A. Curve fitting PERT on histograms

We define an approach to fit a PERT distribution to an execution time histogram. A histogram is a means of categorizing data samples into a number of bins. These bins are identified by a specific range or bin-interval and the width of the interval is the bin-width represented as Δ. In this paper, we consider all bins of a histogram to be of the same width. The normalized height of the bin is given by the number of elements falling within the bin interval divided by the total number of elements in the histogram. We obtain the frequency density by dividing the height of the bin with the bin-width. The number b of bins is a parameter of our approach. Depending on the number of samples, the value of b must be chosen taking into account the trade-off between the fine-graininess of the bin-intervals and the irregularities in the resultant bin heights due to the limited statistical information. The parameters of a PERT distribution are min, max, $mode$ and γ. In case of the execution time distributions, the min and max are set to be the leftmost and the rightmost points, respectively, on the x-axis of the histogram. For the completion time distributions, we use analytically computed bounds as min and max which will be further elaborated in Section VI-C. We compute these bounds analytically since they can be efficiently computed and the simulations are too limited to obtain these extreme values as already explained in Section V-B. With this information, the aim of the curve fitting approach is to derive the $mode$ and γ for the PERT distribution with the best fit on the histogram. In order to define the best fit of one distribution on another, we need a metric that quantifies the fit between two distributions.

Definition 1: (L^2 FUNCTIONS) *A function $f(x)$ is said to be square integrable if $|f|^2 = \int_{-\infty}^{\infty} f(x)^2 dx$ is finite. An L^2 function is a function that is square integrable. In such a case $|f|$ is called its L^2-norm.*

Definition 2: (L^2 INNER PRODUCT) *Given two L^2 functions f and g, their inner product is given by*

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) \cdot g(x)\ dx, \tag{9}$$

Note that $|f|^2 = \langle f, f \rangle$.

Based on the above definitions, we define the curve fitting metric using the normalized inner product of the PERT and the histogram as follows:

Definition 3: (CURVE FITTING METRIC) *Given a PERT distribution p and a histogram h, both L^2 functions, their curve fitting metric ($M_{\langle p,h \rangle}$) is defined by*

$$M_{\langle p,h \rangle} = \frac{\langle p, h \rangle}{|p| \cdot |h|} \tag{10}$$

We normalize the inner product with the L^2 norms of the PERT and the histogram. This is to scale their respective lengths in order to obtain a metric in the range [0,1] that describes how well a PERT curve fits on a histogram. Since the histogram is discrete with a fixed number of bins b, each density value can be obtained using the rectangular function. We use δ_Δ to denote a rectangular function over a bin width Δ and height $\frac{1}{\Delta}$, centered from 0 to Δ.

$$h(x) = \sum_{1 \le k \le b} h_k \cdot \delta_\Delta(x - l_k) \tag{11}$$

where l_k is the left boundary of the k^{th} bin and h_k is its normalized height obtained by dividing the number of the elements in the bin with the total number of elements in the histogram. Given this, the inner product of h and the continuous PERT distribution p is computed as follows.

$$\begin{aligned} \langle p, h \rangle &= \int_{-\infty}^{\infty} p(x) \sum_{1 \le k \le b} h_k \cdot \delta_\Delta(x - l_k) dx \\ &= \frac{1}{\Delta} \sum_{1 \le k \le b} h_k \int_{l_k}^{r_k} p(x) dx, \end{aligned} \tag{12}$$

where l_k and r_k are left and right boundaries of the k^{th} bin. To obtain the curve fitting metric of Equation 10, we divide the inner product of p and h by both their L^2 norms obtained by taking the square root of their respective L^2 inner products. Using Definition 2 and Equation 11, the L^2 inner product of the discrete histogram h with itself can be computed.

$$\begin{aligned} \langle h, h \rangle &= \int_{-\infty}^{\infty} \sum_{1 \le k \le b} h_k \cdot \delta_\Delta(x - l_k) \cdot \sum_{1 \le k \le b} h_k \cdot \delta_\Delta(x - l_k) dx \\ &= \frac{1}{\Delta^2} \sum_{1 \le k \le b} h_k \cdot h_k \cdot \Delta = \frac{1}{\Delta} \sum_{1 \le k \le b} h_k^2 \end{aligned} \tag{13}$$

where Δ is the bin-width of the bins of the histogram.

The L^2 inner product of p with itself can be reduced to the following expression from the PERT equations in Section III (we used Mathematica for the reduction).

$$\langle p,p\rangle = \frac{\Gamma(2\alpha_1-1)\cdot\Gamma(\alpha_1+\alpha_2)^2\cdot\Gamma(2\alpha_2-1)}{(max-min)\cdot\Gamma(\alpha_1)^2\cdot\Gamma(\alpha_2)^2\cdot\Gamma(2(\alpha_1+\alpha_2-1))}, \tag{14}$$

where $\Gamma(n)$ is the Gamma function [4] on n.

B. Divide & conquer search for best fit

In order to find the best fitting PERT distribution, we need to find the $mode$ and γ parameters that give us the highest value for the curve fitting metric with the histogram. We use a divide & conquer search (DCS) approach given in Algorithm 1. The algorithm efficiently searches for a local maximum in a function of two variables on a given interval. The algorithm is used with the function that returns the metric $M_{\langle p,h\rangle}$ for a given histogram h and PERT distribution p with parameters m and γ, where m and γ are the $mode$ and γ parameters. The search converges quickly without requiring a search through all the points in the search space. To compute the best $mode$ and γ combination, a nested divide & conquer search is applied. At the top level, the algorithm searches for the optimal $mode$. To compare different modes, we need to compute the value of the metric corresponding to this $mode$ with its optimal value for γ. So for each chosen $mode$, a second level of divide & conquer search is applied to look for the optimal γ, as given in Algorithm 2. Algorithm 1 takes as input the range for the $mode$ values ([m_{low}, m_{high}]), the range for the γ values ([γ_{low}, γ_{high}]) and the precision up to which we continue the search for the optimal $mode$ and γ denoted as $prec_m$ and $prec_\gamma$. The lower and upper bounds on the $mode$ are the left boundary of the first bin and the right boundary of the last bin, respectively. The lower bound on γ is 4 (default value γ for PERT distributions) and the upper bound is chosen to be a sufficiently large value (10^4 here) that does not exclude the optimum, based on experiments.

Algorithm 1 works by choosing four equidistant $mode$ points (m_1, m_2, m_3 and m_4) covering the given range. When the distance between these points ($interval$) is below the precision $prec_m$ the while loop exits. For each of these four points, the optimal γ and corresponding metric value (M) are computed using Algorithm 2. Algorithm 2 similarly employs a search on the γ parameter. At each of these γ points, with given $mode$, min and max, the value of the metric is computed with the formulae for the curve fitting metric given in Equation 10.

In both algorithms, based on the values of M at these ($mode$ or γ) points, the $selectSegment$ function updates the left and right most points by eliminating at least one and sometimes two of the three intervals based on the values of the metric at each of the points and decides which segment(s) can contain the local maximum assuming uni-modal behavior of the function. For instance, if $M_1 < M_2$, $M_2 > M_3$ and $M_3 > M_4$, it indicates that the curve is increasing between M_1 and M_2 and decreasing between M_2 and M_3 and continues to decrease until M_4. As such the peak cannot lie between M_3 and M_4. The interval is reduced to $[M_1, M_3]$. By continuing the search in this manner, the interval between the points is reduced in every iteration of the loop to at most $2/3^{rd}$ and possibly even $1/3^{rd}$ of its original size. When the while loop exits the search returns the highest metric and the corresponding γ (for Algorithm 2) or $mode$ and γ combination (for Algorithm 1) from the last four points. This algorithm has logarithmic complexity in the size of the interval and converges quickly. It is possible that the histogram being fitted using the DCS algorithm has multiple (local or global) peaks. In this case the algorithm tends to fit a PERT distribution with a $mode$ either around one of the peaks or somewhere in between in such a manner that the overall curve fit is good. The speed of the divide and conquer is preferred over the accuracy of a full search. With respect to γ, we expect the function to be uni-modal; a proof of this conjecture remains to be done.

Algorithm 1: DCS_m

Input : $m_{low}, m_{high}, \gamma_{low}, \gamma_{high}, prec_m, prec_\gamma$
Output: Best fit m, Best fit γ, Best fit M

$m_1 := m_{low}$;
$m_4 := m_{high}$;
$interval := \frac{(m_4-m_1)}{3}$;
while $interval \geq prec_m$ **do**
  $m_2 := m_1 + interval$;
  $m_3 := m_1 + 2 * interval$;
  for *each* $x \in [1,4]$ **do**
    $(\gamma_x, M(m_x)) :=$ $DCS_\gamma(m_x, \gamma_{low}, \gamma_{high}, prec_\gamma)$
  end
  $[m_1, m_4] :=$ $selectSegment(M(m_1,\gamma_1), M(m_2,\gamma_2), M(m_3,\gamma_3), M(m_4,\gamma_4))$;
  $interval := \frac{(m_4-m_1)}{3}$;
end
return $(m_1, \gamma_1, M(m_1,\gamma_1))$

Algorithm 2: DCS_γ

Input : $m, \gamma_{low}, \gamma_{high}, prec_\gamma$
Output: Best fit γ, Best fit M

$\gamma_1 := \gamma_{low}$;
$\gamma_4 := \gamma_{high}$;
$interval := \frac{(\gamma_4-\gamma_1)}{3}$;
while $interval \geq prec_\gamma$ **do**
  $\gamma_2 := \gamma_1 + interval$;
  $\gamma_3 := \gamma_1 + 2 * interval$;
  for *each* $x \in [1,4]$ **do**
    Compute $M(\gamma_x)$ using γ_x and the fixed m
  end
  $[\gamma_1, \gamma_4] :=$ $selectSegment(M(m,\gamma_1), M(m,\gamma_2), M(m,\gamma_3), M(m,\gamma_4))$;
  $interval := \frac{(\gamma_4-\gamma_1)}{3}$;
end
return $(\gamma_1, M(m,\gamma_1))$

C. Obtaining completion time distributions: Combining analysis and simulations

To obtain the completion time distributions, we first analytically compute the min and max points of the PERT distribu-

tions from the bounds of the execution time distributions. Since the schedules are static-order, the best case and worst case completion times can be computed by considering all tasks in their best case and worst case execution times, respectively. In the static order schedule, the fixed execution order of tasks on processors and the processor to processor synchronization mechanism are monotone, i.e., when a task execution time increases, completion times of tasks that follow it in the schedule either remain the same or increase also. Hence, worst-case execution times of tasks are known to lead to worst-case completion times of other tasks and similarly for best-case. As a result, there cannot be any scheduling anomalies allowing for a straightforward computation of the bounds on the completion times of tasks. Once we have the min and max, we draw samples from the PERT execution time distributions and perform simulations to obtain completion time histograms. We then compute the $mode$ and γ values that maximize the curve fitting metric of the PERT on the histogram, using the divide and conquer approach in Section VI-B. Note that the required number of simulations is relatively small, because we only need to estimate the PERT parameters rather than the full distribution, as demonstrated in the experiments section.

D. Robustness metrics

The robustness of a task can be obtained from the probability of missing its deadline. Given the distribution for the completion times of a task A, the probability of missing its deadline d_A is computed as follows.

$$\mathbb{P}[c_A > d_A] = \int_{d_A}^{\infty} p_A^c(t)dt \tag{15}$$

Given the modified PERT completion time distribution of a task A and its deadline d, its probability of deadline miss is the cumulative probability density of the PERT from d to max. It can be computed by substituting the PERT equations from Section III to Equation 15 to obtain the following expression.

$$\mathbb{P}[c_A > d_A] = \int_{d}^{max} \frac{(t-min)^{\alpha_1-1}(max-t)^{\alpha_2-1}}{\beta(\alpha_1,\alpha_2)(max-min)^{\alpha_1+\alpha_2-1}}dt \tag{16}$$

Robustness of the task is complementary to the probability of deadline miss, as given below.

$$R_A = \mathbb{P}[c_A < d_A] = \int_{0}^{d_A} p_A^c(t)dt \tag{17}$$

Given the probabilities of deadline misses per task, we define a random variable X to express the number of tasks that miss their deadline in a schedule. The probability distribution for this random variable is a discrete distribution with probability values for any x tasks missing their deadlines.

$$p(X = x) : \text{Probability that } x \text{ tasks miss their deadlines} \tag{18}$$

The expected value of this random variable gives the expected value of the number of tasks that miss their deadlines in a schedule S. It can be derived by taking the sum of the deadline miss probabilities of its constituent tasks. Note that this also applies if the completion time distributions of the tasks are dependent.

$$EX = \sum_{A \in T} \mathbb{P}[c_A > d_A] = \sum_{A \in T} (1 - R_A) \tag{19}$$

Normalizing the expected value with the number of tasks in S and taking the complement is defined as a metric for the robustness of S.

$$R_S = 1 - \frac{\sum_{A \in T}(1 - R_A)}{|T|} \tag{20}$$

The following section gives the experimental results obtained by applying this approach on real schedules.

VII. Experiments and results

We performed robustness analysis on the schedules of three critical applications of the latest innovation of ASML wafer scanners, the TWINSCAN NXE 3300B [1]. These lithography machines project the patterns of electronic circuits stored on reticles to flat silicon wafers with a precision of $22nm$ by exposing them to extreme ultraviolet light. The part of the machine where the exposure happens is called the wafer stage. It consists of chucks that are elements which hold the wafers and move under the light source with extreme accuracy to ensure the high resolution and overlay. One of the three applications analyzed is the *Wafer stage*, explained below. The two other applications we studied are the *Wafer handler* and *Imaging control*. We chose to present the Wafer stage in detail due to bigger size and higher complexity. The other two application gave us similar results. We computed PERT execution time distributions for tasks, based on available measurement statistics. We then performed robustness analysis by drawing samples from these distributions, simulating completion time histograms, and fitting PERT distributions with analytically computed bounds. The simulations were performed by converting the schedule models to discrete event simulation models in POOSL [20] and run with the Rotalumis simulation tool.

The wafer stage application is involved in ensuring the high speed movements of the wafer stage chucks and other components according to a motion profile in the nanometer range. It is a complex network of several communicating subsystems which control the movement of the chucks and exposure of the wafer. Its schedule consists of 4219 tasks running at a frequency of $10kHz$ on a platform consisting of 11 general purpose processors. We analysed the actual schedules on 80% of the processor budget, targeting the analysis to future high performance machines. This gives the tasks a deadline of $8 \cdot 10^{-5}s$. There are also 22 critical tasks that send data to actuators; they have a much smaller deadline ranging from 40% to 45% of the processor budget translating to between $4 \cdot 10^{-5}$ to $4.5 \cdot 10^{-5}s$.

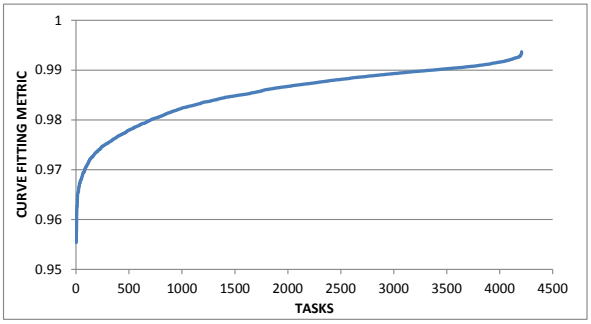

Fig. 3: Curve fitting metrics for all tasks of the TWINSCAN NXE wafer stage application.

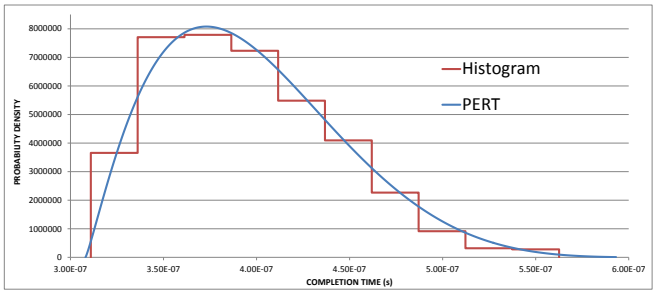

Fig. 4: PERT fit on a histogram with sufficient mass at the extremes.

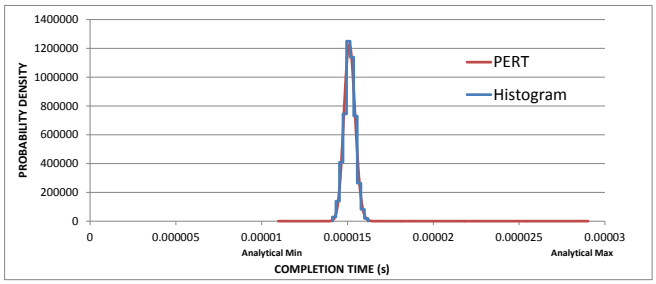

Fig. 5: PERT fit on a histogram with missing rare events.

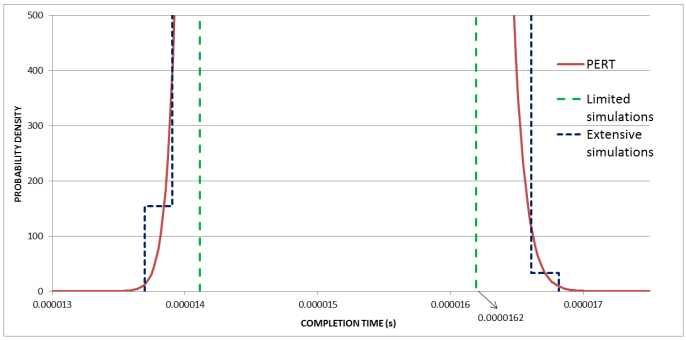

Fig. 6: Comparison of PERT fit on limited and extensive simulations.

A. *Evaluation of the robustness analysis approach*

Simulating 1000 samples for the wafer stage schedule takes approximately $240s$. The nested divide and conquer search takes approximately $0.1s$ per task. The value of the curve fitting metric obtained for all tasks of the wafer stage schedule is summarized in Figure 3. We can see that all tasks have a good curve fit value M of above 94%. This shows that PERT is a good distribution for the completion times. To visualize how the PERT fitting is able to approximate mass around the rare events, we have also shown the PERT fit on the histograms of two tasks. Figure 4 shows a task with sufficient mass in the histogram to produce an accurate fit. On the other hand Figure 5 shows a task with missing mass around the rare events which is covered by the PERT distribution. The robustness of the wafer stage schedule, in terms of the expected value of the number of tasks that miss their deadlines from Equation 19, was found to be around 79 (which is around 2% of the tasks). The average of the number of tasks that miss deadlines in all the simulation runs was found to be around 78 with a variance of around 133 and standard deviation of around 11, which is close to the predictions. However, the simulations are not sufficient to accurately estimate robustness per task, and our method to assess schedule robustness is much faster than extensive simulations, as explained in the next subsection.

B. *Validation with extensive (day-long) simulations*

In order to validate our results we performed extensive simulations (24 hours) for the completion time of the task from Figure 5. We placed the histogram obtained from the extensive simulation on top of the PERT distribution fitted on the histogram of a limited simulation of 1000 runs from Figure 5. We consider a snapshot by zooming into the bottom portion to observe the outer bins of the histogram as shown in Figure 6. Note that the inner vertical lines are the outer edges of the outermost bins of the histogram from the limited simulation. Figure 6 makes it clear that the extensive simulation produced results outside of the bins of the histogram of the limited simulation. The probability of deadline miss assuming, for the sake of the example, a deadline of $16.2\mu s$ is 0 from the limited simulation and $7.3 \cdot 10^{-4}$ from the PERT distribution. On the other hand, the probability of deadline miss at $16.2\mu s$ is approximated as $9.1 \cdot 10^{-4}$ from the extensive simulation. This shows that the PERT derived from the limited simulation gives reasonable estimates for events that only occurred in the extensive simulation. The values of the curve fitting metric of the PERT on the extensive simulation is 0.9949 and on the limited simulation is 0.9925, which confirms that the PERT obtained from the limited simulation fits even better on the extensive simulation results. Hence, we observe that PERT is a good distribution to perform robustness analysis of a task, which cannot be done accurately with only limited simulations. From these results, we can conclude that the proposed approach is sufficiently fast to be practically useful,

whereas using extensive simulations to obtain statistically relevant results is not practically feasible.

VIII. Conclusions and discussions

In this paper, we have presented an approach to perform robustness analysis of multiprocessor schedules. The overall approach addresses the complexity of performing max-plus operations on distributions by using a combined analytical and limited simulations-based approach. It uses a new curve fitting approach to derive the distributions needed for this analysis. The metrics that we obtain quantify the robustness of tasks and schedules. Our future aim is to approximate this analysis by using incremental simulations to reduce its run time even further in such a way that it can be incorporated within the scheduling loop. It can then be used to make decisions that steer the scheduler towards achieving more robust schedules. We also intend to study the effect of reducing the number of simulations on the accuracy of the results and perform a trade-off analysis between accuracy of results and computation times.

References

[1] ASML. http://www.asml.com. Accessed: 28/02/2014.

[2] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Computer Aided Design, ICCAD-2003. International Conference on*, pages 900–907, 2003.

[3] S. Ali, A. Maciejewski, H. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *Parallel and Distributed Systems, IEEE Transactions on*, 15(7):630–641, July 2004.

[4] E. Artin. *The gamma function*. Holt, Rinehart and Winston, New York, 1964.

[5] M. Berkelaar. Statistical delay calculation, a linear time method. In *TAU (ACM/IEEE workshop on timing issues in the specification and synthesis of digital systems)*, December 1997.

[6] S. Bhardwaj, S. Vrudhula, and D. Blaauw. τau: Timing analysis under uncertainty. In *Computer Aided Design, ICCAD-2003. International Conference on*, pages 615–620, 2003.

[7] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: From basic principles to state of the art. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(4):589–607, April 2008.

[8] L. Boloni and D. C. Marinescu. Robust scheduling of metaprograms. *Journal of Scheduling*, 5(5):395–412, 2002.

[9] L.-C. Canon and E. Jeannot. Evaluation and optimization of the robustness of dag schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):532–546, 2010.

[10] C. E. Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145–162, 1961.

[11] D. England, J. B. Weissman, and J. Sadagopan. A new metric for robustness with application to job scheduling. In *HPDC*, pages 135–143. IEEE, 2005.

[12] P. E. Greenwood and M. S. Nikulin. *A guide to chi-squared testing*. Wiley-Interscience, 1996.

[13] S. Nadarajah and S. Kotz. Exact distribution of the max/min of two gaussian random variables. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(2):210–212, Feb 2008.

[14] A. Renyi. On the central limit theorem for the sum of a random number of independent random variables. *Acta Mathematica Academiae Scientiarum Hungarica*, 11(1-2):97–102, 1963.

[15] R. Rubinstein and D. Kroese. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Statistics. Wiley, 2008.

[16] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel. Stochastic robustness metric and its use for static resource allocations. *J. Parallel Distrib. Comput.*, 68(8):1157–1173, Aug. 2008.

[17] Z. Shi, E. Jeannot, and J. Dongarra. Robust task scheduling in non-deterministic heterogeneous computing systems. In *Cluster Computing, 2006 IEEE International Conference on*, pages 1–10, 2006.

[18] F. Stork. *Stochastic resource-constrained project scheduling*. PhD thesis, Technische Universität Berlin, 2001.

[19] B. Theelen. *Performance Modelling for System-Level Design*. PhD thesis, Eindhoven University of Technology, 2004.

[20] J. P. M. Voeten, P. van der Putten, M. Geilen, and M. P. J. Stevens. Formal modelling of reactive hardware/software systems. In *J.P. Veen, Ed., Proceedings of ProRISC/IEEE 97, Utrecht : STW, Technology Foundation*, pages 663–670, 1997.

[21] D. Vose. *Risk analysis : a quantitative guide*. Wiley, Chichester, England, Hoboken, NJ, 2008.

[22] D. Wang, B. Gong, and G. Zhao. Estimating deadline-miss probabilities of tasks in large distributed systems. In R. Li, J. Cao, and J. Bourgeois, editors, *Advances in Grid and Pervasive Computing*, volume 7296 of *Lecture Notes in Computer Science*, pages 254–263. Springer Berlin Heidelberg, 2012.