On Robust Task-Accurate Performance Estimation

Yang Xu  Bo Wang  Ralph Hasholzner
Intel Mobile Communications
Munich, Germany
{yang.a.xu, bo1.wang, ralph.hasholzner}@intel.com

Rafael Rosales  Jürgen Teich
University of Erlangen-Nuremberg
Erlangen, Germany
{rafael.rosales, teich}@informatik.uni-erlangen.de

ABSTRACT
Task-accurate performance estimation methods are widely applied in early design phases to explore different architecture options. These methods rely on accurate annotations generated by software profiling or real measurements to guarantee accurate results. However, in practice, such accurate annotations are not available in early design phases due to lack of source code and hardware platform. Instead, estimated mean or worst-case annotations are usually used, which makes the final result inaccurate because of the errors induced by the estimations, especially for designs with tight time constraints. In this paper, we propose a novel methodology that combines Distributionally Robust Monte Carlo Simulation with task-accurate performance estimation method to guarantee robust system performance estimation in early design phases, i.e., determining the lower bound of the confidence level of fulfilling a specific time constraint. Instead of using accurate annotations, our method only uses estimated annotations in the form of intervals and it does not make any assumptions of the distribution types of these intervals.

Categories and Subject Descriptors: B.8.2 [Performance and Reliability]: Performance Analysis and Design Aids

General Terms: Performance

Keywords: task-accurate, robust performance estimation, distributionally robust Monte Carlo simulation

1. INTRODUCTION
In early System-on-Chip (SoC) design phases, to evaluate different system architectures, high-level performance estimation methods are usually applied during the exploration of the design space. Correct design decisions made in early design phases are very important for avoiding significant modification efforts and cost in later phases. Therefore, robust early performance estimations are required in order to take correct early design decisions. Additionally, owing to the increasing complexity of modern SoCs, the design space becomes very huge. Thus, fast performance estimation methods are mandatory to allow for an efficient design space exploration.

To achieve such a high efficiency during design space exploration, task-accurate performance estimation methods have been proposed [5, 13, 15]. In these methods, basic system operations, such as functions or communication transactions, are modeled as tasks. The execution time of each task is back-annotated into the task model for fast performance estimation through either simulation or model-based analysis. Since a task is the finest granularity in such kind of method, it is called Task-Accurate Performance Estimation (TAPE). In [5], the execution time of each task is modeled by annotating the delay budgets to the communication events. Then, a virtual processing unit is introduced to simulate the timing behavior of executing the tasks on the corresponding system. Thanks to its XML-based performance model, the design space exploration is significantly accelerated. Similarly, in [15], an action-accurate virtual processing component approach is proposed. It differentiates itself from [5] by applying an actor-oriented modeling approach [9] [10]. By strictly separating data flow from control flow within each actor, timing may be annotated individually to the actions instead of a full task, given a finer timing granularity and thus precision during timing analysis. A hybrid method is proposed in [13] where Worst-Case Execution Time (WCET) analysis is first performed and the resulting WCET values are back-annotated to the task model to simulate system performance. Dynamic timing variations, such as caused by data dependencies, are also incorporated by proposing dynamic correction. All these methods can simulate system performance very efficiently. However, none of them can guarantee a robust early performance estimation because in early design phases the annotation values these methods rely on are usually inaccurate. With the term robust estimation, we designate the estimation methods that enable a determination of the error and error bounds of a performance estimation, e.g., using confidence level. In early design phases where software and hardware might not yet be available for generating accurate annotation values, estimated timing information must be used to estimate system performance. This may make the final estimation result inaccurate and unreliable. Even though in early design phases relatively accurate results can still help the designers differentiate different architecture options, robust performance estimation is still mandatory when critical design decisions or tight time constraints are involved. For example, given two baseband processors architectures arch1 and arch2, the TAPE methods can only confirm that arch1 is faster than arch2 in processing Long Term Evolution (LTE) packets. But because of the inaccuracy of the performance estimation they cannot confirm whether these two architectures can finish the packet processing within 1 ms (real-time constraint of LTE) especially when the estimated time is too close to the real-time constraint. Therefore, it is highly desired to have a robust performance estimation method providing at least some confidence information, i.e., the probability of an architecture meeting a specific performance constraint, given errors existed in the estimation results.

A few low-level methods have been proposed to estimate performance with confidence information, e.g., [2, 4]. In [4], the software behavior is modeled by a sequence of virtual instructions with each one representing a type of real instructions. Then linear regression methods are used to determine a predictor equation, which estimates the resulting performance along with confidence levels. The drawback of this method is that the statistical predictor equation is only trained for a specific application population; whenever a new type of application is under evaluation, it needs a new dedicated equation. To overcome this limitation, the work in [2] proposes a trace-based estimation method. Instead of training a statistical equation...
1. We model the TAPE result as a function of statistical variables whose exact distribution types may be unknown and we prove that DRMCS can be applied on TAPE to extract robust system performance estimation, i.e., performance estimation with worst-case confidence levels of fulfilling specific time constraints.

2. Instead of accurate timing annotations, which are usually unavailable in early design phases, our method only requires estimated annotations, which are specified by intervals;

3. Our method complements traditional TAPE methods by adding the capability to provide worst-case confidence information on meeting a performance constraint. This information may therefore guide designers to explore architecture options with respect to real-time constraints.

The rest of this paper is organized as follows: In Section 2 we introduce the motivation of our work and some basics of DRMCS. In Section 3 we present our robust task-accurate performance estimation methodology in details. Thereafter, experimental results are given in Section 4 and the paper is concluded in Section 5.

2. MOTIVATION & PRELIMINARIES

In this section, we will first present the motivation of our work. Then, we will introduce some preliminary knowledge of DRMCS, a known statistical method that will be used in our methodology.

2.1 Motivation

As mentioned previously, in order to predict the performance of a new design in early SoC design phases, estimated execution time of tasks must be used to execute TAPE simulations. These estimated values may be generated by extrapolation, taking measurements from previous products as a baseline and projecting the new execution time based on the differences between two generations and sometimes even based on the experience of the designers. Such estimated values are usually in the form of intervals, e.g., the decoding time of 1 byte of MP3 data is specified as 220 ns ± 20 ns on a 312 MHz processor. The exact distribution type of such intervals are usually unknown. Since traditional TAPE typically relies on constant annotation values, mean values of such intervals, e.g., 220 ns in this case, are used to simulate average system performance, which does not cover all the possible cases. Therefore, such kind of results cannot be used to guide designs with tight absolute time constraints. Additionally, corner-based worst-case execution time analysis is usually too pessimistic and will lead to over-engineering.

To solve this practical problem, we propose a statistical method that can make good use of the estimated intervals to generate robust system performance estimation. In this method, the execution time of each task is modeled as a mean value + a random variable representing the estimated intervals whose exact distribution types may be unknown. The expected results of this method are performance estimations with worst-case confidence levels of fulfilling specific performance constraints.

2.2 DRMCS

In statistics, such kind of problem has been tackled by proposing Distributionally Robust Monte Carlo Simulation (DRMCS) [1, 6-8]. DRMCS first evaluates whether the set of so-called good values of the random variables is of specific shapes, e.g., convex. In our context, an example of such good values would be those resulting in the fulfillment of a time constraint. If positive, then it can efficiently estimate the worst-case probabilistic performance metric such as the confidence level of meeting a time constraint, given the intervals and the class of distributions of the variables. The difference between DRMCS and the traditional Monte Carlo (MC) simulation is that instead of requiring the exact distribution types of the variables, DRMCS only needs the class of distributions of the variables and can still find the worst-case value of the probabilistic performance metric.

2.2.1 Notations

Here we first introduce some basic notations related to DRMCS.

1. Uncertain parameter space: considering a system with uncertain parameters \( Q = [q_1, q_2, ..., q_L]^T \in \mathbb{R}^L \), radius \( r \in [r_1, r_2, ..., r_L]^T \), we define the uncertain parameter space as

\[
Q_r = \{ Q : |q_i| \leq r_i, i = 1, 2, ..., L \}
\]

Given bounds \( |q_i| \leq r_i \) for \( i = 1, 2, ..., L \). In our context, \( q_i \) correspond to the random variables representing the estimated intervals and \( r_i \) are the ranges of the intervals.

2. Probabilistic performance metric: we define the probabilistic performance metric as the probability of fulfilling a performance constraint \( \gamma \)

\[
\Phi(f) = \text{Prob}(\phi(Q) \leq \gamma)
\]

where \( \phi(Q) \) is the deterministic performance metric and \( Q \) means the joint probability function of \( Q \) is \( f \). For example, the probability that the execution time is smaller than \( \gamma \) assuming the random variable vector \( Q \) has the statistical distribution \( f \).

3. Good value space: we define the good value space as

\[
Q_g = \{ Q \in Q_r : \phi(Q) \leq \gamma \}
\]

One example of \( Q_g \) in our context would be all timing annotation value samples making the performance constraint \( \gamma \) fulfilled.

4. Worst-case probabilistic performance metric is defined as

\[
\hat{\Phi}(f) = \min_{\forall f \in \mathcal{F}} \text{Prob}(\phi(Q) \leq \gamma) = \min_{\forall f \in \mathcal{F}} \text{Prob}(Q' \in Q_g)
\]

where \( \mathcal{F} \) is the class of distribution of the random variables and \( f \in \mathcal{F} \) is the distribution type that minimizes \( \hat{\Phi}(f) \). In our case, \( \hat{\Phi}(f) \) is a worst-case confidence level of fulfilling a specific performance constraint.

2.2.2 Principles of DRMCS

The key point of DRMCS is that with respect to certain classes of distributions, the worst-case probabilistic performance metric \( \hat{\Phi}(f) \) is obtained when \( f \) is the uniform distribution [1, 6-8]. Additionally, it turns out that different classes of distributions and the shape of \( Q_g \), e.g., convex or unirectangular, also have a large impact on achieving \( \hat{\Phi}(f) \). In the following, we will introduce the DRMCS principles for the most interesting class of distributions.

Class of Independent Symmetric Distributions \( f_{\text{sym}} \): requires
1. the probability density function (pdf) \( f \) of each variable must be symmetric and nonincreasing with \( |q_i| \); 2. the uncertain parameters \( q_1, q_2, ..., q_L \) must be independent, i.e.,

\[
f(Q) = \prod_{i=1}^{L} f(q_i)
\]

Many distributions can fall into this class,
The mathematical form, which can be directly evaluated against the
First transform the TAPE simulation into a task graph ex-
robust performance estimation at the system level.
we will show that DRMCS can also be used to estimate robust
approaches for the three classes of distributions. In this paper,
partial statistical information on process variations by showing
TAPE problems. In [17], the authors prove that the DRMCS
need first to prove that the DRMCS principles are applicable to

3. ROBUST TASK-ACCURATE PERFORMANCE ESTIMATION
In order to use DRMCS to extract robust TAPE results, we
need first to prove that the DRMCS principles are applicable to
TAPE problems. In [17], the authors prove that the DRMCS
method can be applied to estimate robust timing yield with
partial statistical information on process variations by showing
approaches for the three classes of distributions. In this paper,
we will show that DRMCS can also be used to estimate robust
TAPE results with estimated annotations, and hence provide
robust performance estimation at the system level.

3.1 Task Graph Expression of TAPE Simulation
To prove the TAPE problem meets the conditions of DRMCS, we
first transform the TAPE simulation into a task graph ex-
pression [16] [3] and then formulate the simulation result in a
mathematical form, which can be directly evaluated against the
DRMCS conditions.
A task graph is a directed acyclic graph with nodes represent-
ting tasks and edges representing data dependencies between
tasks, as shown in Fig. 1. Each task is an atomic operation
in the system. It only starts its execution when all its input
are available and releases its outputs after the execution
terminates. This level of abstraction is a perfect match for
the specification of TAPE simulation, which is equivalent to a
repeated execution of the functionality described by the task
graph. The raw execution time of each task is determined by
the hardware component it is mapped to. In TAPE, it usually
exists in the form of annotation values and it is independent of
the token data passed between tasks.
The TAPE result (e.g., total execution time) is equivalent to the
latency of the Longest Execution Path (LEP) that crosses
the first started task and the last terminated task. However, the
LEP cannot be directly obtained from the task graph, since the
task graph does not contain any resource sharing dependency
information, which, in addition to data dependencies, also in-
fuences the LEP. For example, if task p3 and task p4 in Fig. 1
are mapped to the same hardware resource, e.g., a processor,
even though there is no data dependency between them, they
cannot be executed in parallel due to the resource sharing.
This means in order to obtain the LEP, such kind of tasks must
be serialized in the task graph. Therefore, we define a Serialize
operation to incorporate the resource sharing dependency.

Serialize operation. If task pi and task pj, i j are mapped to
the same resource and there is no data dependency between
them, they are serialized by adding a resource dependency edge
dot edge in Fig. 2 ) between them. The direction of the edge is
determined by the scheduling algorithm assigned to the resource.

In Fig. 2, we assume p3, p4, p6, p7, p8 are mapped to the same
resource. By applying the serialize operation, the LEP can be
easily obtained from the serialized task graph, namely, p0 p1 p3 p4 p5 p6 p7 p8 assuming all the tasks
have the same execution time. Note that some of the resource
dependency edges are redundant and can be removed, e.g., the
one from p3 p4.

After the task graph transformation, we can start to formu-
late the TAPE simulation result in a mathematical form. We
model the execution time of each task as a mean value + a
random variable, e.g., for task p1 we have

where D0 is the mean value of the execution time of pi, qi
represents the Errors Induced by Estimation (EIE), N is number
of tasks in the system and ai are constant coefficients.
For an arbitrary path k in the serialized task graph, the path
delay is

where \( D_k \) represents the errors induced by estimation (EIE), \( N \) is the number of tasks in the system and \( A_i \) are constant coefficients.

For an arbitrary path \( k \) in the serialized task graph, the path
delay is

\[
D_k = \sum_{i \in Path(k)} D_i = \sum_{i \in Path(k)} D_{0i} + \sum_{i \in Path(k)} a_i q_i = D_0 + A^k Q
\]

where \( Q = [q_1, q_2, ..., q_M]^T \), \( A^k = [a_1, a_2, ..., a_M] \), \( M \) is the
number of tasks on path \( k \).
Now, the delay of the LEP can be computed as

\[
D_{LEP} = \max(D^1, D^2, ..., D^N)
\]

where \( P \) is the number of paths in the serialized task graph.
For a given time constraint \( D_i \), the confidence level of fulfilling
this constraint is defined as

\[
CL(D_i) = \text{Prob}(D_{LEP} \leq D_i)
\]
If the exact pdf of the EIE $q_i$ is known, we can calculate the exact $CL(D_i)$ for a specific time constraint. However, in early design phases, we only have partial statistical information, such as the intervals of $q$. This is a similar problem as in DRMCS in Section 2. By applying the DRMCS method, we can obtain the worst-case confidence level $CL(D_i)$ if the pdf of $q_i$ belongs to a class of distributions $\mathcal{F}$

$$CL(D_i) = \min_{Q \in \mathcal{F}} CL(D_i)$$  \hfill (11)

where $\mathcal{F}$ is the class of distributions such as $\mathcal{F}^{SI}$, $\mathcal{F}^{A1}$ and $\mathcal{F}^{SD}$, as mentioned in Section 2. Note that the mathematical formulae are only used to evaluate against the DRMCS conditions. The performance is still estimated per simulation. In the following we will match the pdf of EIE to one of these classes and derive the principle for robust TAPE.

### 3.2 Robust Confidence Level for TAPE with EIE

We assume the random variables $q_i$ representing the EIE are independent, symmetric and nonincreasing with $|q_i|$. This assumption is reasonable because 1) in early design phases, the execution time of each task is estimated independently so that the errors induced by estimation, the EIE, are also independent of each other; 2) the symmetry assumption suggests that the positive and negative deviations from the mean execution time are equally likely; 3) the small estimation errors are more likely than large errors. Therefore, they are nonincreasing with $|q_i|$. Based on this assumption, we can category the pdf of EIE as the $\mathcal{F}^{SI}$ class in Section 2.

For class $\mathcal{F}^{SI}$, the Princetown principle can be applied without any restrictions. To apply the Unimodality principle under class $\mathcal{F}^{SI}$ for efficient robust estimations, we need to prove that the $Q_o$ of TAPE is convex or unirectangular (6).

**Lemma 1.** The shape of $Q_o$ of TAPE is convex with respect to EIE $Q$ when the path delay is given by (8) and the LEP delay is given by (9).

**Proof.** According to the definition of $Q_o$, the $Q_o$ of TAPE is defined as $Q_t^{TAPE} = \{Q \in Q_i: D_{lep} \leq D_t\}$. In order to prove the shape of $Q_o$ of TAPE, we first prove that the shape of the $Q_o$ of a single path is convex. Based on the definition of convexity [11], we only need to show that for any two EIE vector $Q_i$, $Q_j$, $i \neq j$, if these two vectors result in a fulfillment of the time constraint, the new vector $\alpha Q_i + (1 - \alpha)Q_j$, $\alpha \in [0, 1]$, also results in meeting the constraint.

According to (8), for any two vectors $Q_i, Q_j \in Q_o$, $Q_k \neq Q_o, i \neq j$. By definition of $Q_o$, we have

$$D_o^k + A^Q Q_i \leq D_t \\forall k = 1, 2, ..., P$$  \hfill (12)

$$D_o^k + A^Q Q_j \leq D_t \\forall k = 1, 2, ..., P$$

where $Q_i = [0, 0, ..., q_i, ..., 0]^T, Q_j = [0, 0, ..., q_j, ..., 0]^T, i \neq j; P$ is the number of paths in the system. This is equivalent to

$$A^Q Q_i \leq D_t - D_o^k \iff A^Q Q_j \leq \alpha (D_t - D_o^k)$$

$$A^Q Q_i \leq D_t - D_o^k \iff A^Q (1 - \alpha)Q_j \leq (1 - \alpha)(D_t - D_o^k)$$

When we sum up the two equations above, we obtain

$$A^Q [\alpha Q_i + (1 - \alpha)Q_j] \leq (D_t - D_o^k)$$

$$A^Q [\alpha Q_i + (1 - \alpha)Q_j] \leq (D_t - D_o^k) \\forall k = 1, 2, ..., P$$  \hfill (14)

which means $\alpha Q_i + (1 - \alpha)Q_j \in Q_o$ and this proves that the shape of $Q_o$ is a single path is convex. Because the delay of the LEP is the maximum delay of all the paths and the max operation preserves convexity [11], we can finally conclude that the shape of $Q_t^{TAPE}$ is also convex. This completes the proof.

Now, we can apply the Uniformity principle for efficient robust TAPE with worst-case confidence levels, which requires only a single MC simulation $^1$ with each random variable $q_i$ uniformly distributed in $[-r_i, r_i]$. The confidence level of fulfilling a specific time constraint is obtained by calculating the probability of the total execution time less than the constraint. The DRMCS method guarantees that this confidence level is the worst-case one. Within each iteration of MC simulation, one traditional TAPE simulation with sampled annotation values is executed. Since a single TAPE simulation is very efficient and the number of tasks are usually quite small, usually less than 100, the whole simulation can be finished very fast, as shown in the experimental results. Additionally, Latin Hypercube Sampling (LHS) [12] can be used to further accelerate the simulation. Note that our method is not restricted to simulation-based performance estimation approach; it can be easily applied to analytic approach by substituting the TAPE simulation inside each MC iteration with model-based analysis.

### 3.3 Frequency Scaling Induced Dependency

Dynamic Voltage & Frequency Scaling (DVFS) is widely applied on modern SoC chips for dynamic power management. This technique is also modeled in some TAPE framework [18] to estimate power and performance jointly. Modeling DVFS in TAPE makes the robust TAPE even more complex, since it induces the frequency scaling dependency. For example, a task $p$ is mapped to a CPU that supports two frequency settings, a fast mode (1 GHz) and a slow mode (500 MHz). In TAPE, two annotations may therefore be assigned to the same task $p$ in the two speed modes, $p_{fast}$ and $p_{slow}$, since it has different execution time in these modes. Apparently, the EIE for $p_{fast}$ and $p_{slow}$ are dependent on each other, e.g., if $p_{fast}$ has a positive error, the $p_{slow}$ should have a positive error, too. This frequency scaling induced dependency therefore violates our previous assumption of independent variables, which makes the Uniformity principle inapplicable. Fortunately, the EIE for $p_{fast}$ and $p_{slow}$ are fully correlated, i.e., if the execution time of $p_{fast}$ is 1 ms, the counterpart of $p_{slow}$ must be 2 ms due to the frequency scaling factor. Therefore, when we formulate the delay of a path in a mathematical form, the random variables for the same task in different frequency modes can be merged together $^2$ and the merged variable is still independent of the variables of other tasks. This makes sure that the Uniformity principle can still be used.

In our implementation, this dependency is incorporated by a linked sampling method. This means we only sample for the annotations of tasks in the fastest mode and take the sample values multiplied by a frequency scaling factor as the samples for a slower mode.

### 4. EXPERIMENTAL RESULTS

We applied the proposed methodology to a real world scenario, namely, MP3 decoding time estimation, to evaluate its effectiveness. We choose an actor-oriented TAPE framework [18] as the base of our robust TAPE simulation. To further accelerate the simulation, we applied LHS and parallelized MC simulation at the same time. The simulations were executed on a Linux machine with 4 GB RAM and a 3.4 GHz CPU, which has 4 cores and each core supports 2 threads executing in parallel.

$^1$A single MC simulation means one MC simulation with a series of iterations.

$^2$It is guaranteed that they are on the same path because they share the same resource.
4.1 Robust TAPE vs Traditional TAPE

We first compare our method with the traditional TAPE with worst-case annotations to show its advantages during design space exploration. We modeled a simplified embedded platform which consists of one CPU, one DSP, memory subsystem, interconnections and a dedicated hardware, as illustrated in Fig. 3. The AFE (Audio Front End) is an analog hardware responsible for decoded data playback. The functionality of MP3 decoding is modeled by a data flow graph as shown in Fig. 4, which is composed of five actors, namely, source (Src), pre-processing (PrePro), decoding (Dec), post-processing (PostPro) and playback (Play). The MP3 decoding time is defined as the time between the data streaming into the PrePro stage and the data leaving the PostPro stage. Both CPU and DSP support two frequency settings, fast and slow. Whenever their computing load exceeds a specific threshold, they are switched to the fast mode automatically. The actor mappings, execution time intervals and frequency scaling factor \( s = \frac{f_{\text{fast}}}{f_{\text{slow}}} \) are specified in Table 1, part I. We assume the uncertain radius \( |r_i| \) is 10% of the mean execution time for all tasks. Since the Src models the data source, it has no real corresponding processing unit and its execution time is annotated as 0 us. Note that the communication latencies can be easily modeled by adding communication tasks between any pair of tasks and considering their execution time, too.

We first use our robust method to estimate the time of decoding 2 KB MP3 data. Then we run the traditional TAPE simulation with annotation values being the upper bound of the intervals (worst case). Fig. 5 shows us the Cumulative Distribution Function (CDF) of the decoding time estimated by our robust method (labeled as Robust), from which we can see that the estimation reaches 100% confidence level at around 28.2 ms. However, the execution time estimated by the traditional TAPE with worst-case annotations is 28.53 ms, which is apparently too pessimistic. This pessimism will become even worse if the \( |r_i| \) are larger. Therefore, we can conclude that in spite of the unavoidable EIE of annotation values, our method can still provide more confident and realistic estimation than traditional mean and worst-case based TAPE.

In order to evaluate the robustness, we compare the CDF generated by our method with the ones generated by normal MC simulations with the distribution types of the intervals being Gaussian and Triangular. The comparison is also illustrated in Fig. 6, which confirms that both Gaussian and Triangular predict more optimistic confidence levels; our method guarantees the worst-case confidence level.

We also found that the confident decoding time cannot be simply obtained by adding a margin to the mean decoding time even though all the tasks have the same margin (10% in this case), because different tasks contribute differently to the total execution time, thus they have different EIE sensitivities. We analyze the sensitivities of the tasks by computing the Standardized Regression Coefficients for each task and found that PostPro has the highest sensitivity while PrePro has the lowest. To verify this, each time we artificially improve the EIE of one task by 5% and rerun the simulation. Then we compare the confidence level improvement of each task, as shown in Fig. 6.

The CDF marked with PrePro is nearly the same as the old CDF, which indicates the confidence level remains the same although the EIE of PrePro has been improved; while the CDF marked with PostPro is improved significantly. This suggests that in early design phases we can use our framework to analyze the sensitivity of tasks with respect to timing guarantees and improve the annotations of the tasks with high EIE sensitivity to effectively improve the confidence level of the estimations.

4.2 Architecture Exploration with Respect to Time Constraints

In the following we will describe a case study to demonstrate the effectiveness of our method to explore different architectures with respect to real-time constraints. In this case study, we would like to reduce the cost of the platform in Fig. 3. One architecture option would be to replace the processors with slower ones (5 times slower), noted as cpu-dsp. And another more aggressive option is to even get rid of the DSP by mapping all actors to the CPU, noted as all-cpu. Before making these design decisions, we need to evaluate whether these new architectures will meet a given latency, respectively throughput constraint for processing a chunk of 2 KB MP3 data.
ration of 1 : 11 for 2 KB MP3 data, the decoding must be finished within $2048 \times 11)/44.1 = 127$ ms. The mappings, execution time intervals and frequency scaling factor $s$ of these new architectures are specified in TABLE 1, part II.

Fig. 7 shows the CDFs of the decoding time for both cpu-dsp and all-cpu architectures. According to the mean decoding time estimated by the traditional TAPE (110.1 ms, 121.5 ms), both architectures meet the time constraint (127 ms). However, the confidence levels tell us that only the cpu-dsp architecture can meet the real-time constraint with nearly 100% confidence while the all-cpu architecture can meet the deadline with only about 80% confidence. Therefore, we can confirm that the cpu-dsp architecture can be applied to lower the cost and it would be very risky to use the all-cpu architecture (20% chance to miss the deadline). From this case study, we can conclude that applying traditional TAPE to guide designs with tight time constraints may lead to a constraint violation; in contrast our method is able to provide the worst-case confidence levels of fulfilling time constraints, which can be safely used to guide time constrained designs despite of the existence of EIE.

### 4.3 Simulation Time

It took a quad-core CPU 29.67 s to finish one robust estimation. Since there are no general rules for the sample size of LHS, we choose a conservative sample size, i.e., 2000 samples, for our simulation, which makes the ratio of sample size and variables (2000/3) much larger than the ones used in literature [14] (10-20). This conservative sample size guarantees convergence of the results, as shown in Fig. 8 where the CDF of LHS with 2000 samples matches very well with the CDF of MC simulation with 20000 samples. It also indicates that the simulation speed of our method still has a lot of improvement potential.

### 5. CONCLUSIONS

In this paper, we propose a novel robust task-accurate performance estimation method that only needs estimated annotation values to generate robust worst-case timing confidence levels, which can be used to guide the search of designs satisfying tight time constraints. The DRMCS method is applied within this method to guarantee an estimation with worst-case confidence level of fulfilling a specific time constraint. Experimental results confirm that our method can provide more reliable and realistic estimations than the traditional mean value based TAPE methods; in early design phases, our method can be used to win confidence in the fulfillment of time constraints during exploration of time-critical architecture and mapping decisions.

### Acknowledgments

This work was supported in part by the Project PowerEval (funded by Bayerisches Wirtschaftsministerium, support code IUK314/001).

### 6. REFERENCES


