A Fast and Highly Accurate Statistical Based Model for Performance Estimation of MPSoC on-Chip Bus

FARHAN SHAFIQ, TSUYOSHI ISSHIKI, DONGJU LI, HIROAKI KUNIEDA
(farhanshafiq,isshiki,dongju,kunieda)@vlsi.ce.titech.ac.jp
Dept. Communications and Computer Engineering, Tokyo Institute of Technology, Tokyo, Japan.

***Abstract*—While Multiprocessor System-On-Chips (MPSoCs) are becoming widely adopted in embedded systems, communication architecture analysis for MPSoCs becomes ever more complex. There is a growing need for faster and accurate performance estimation techniques for on-chip bus. In this paper, we present a novel statistical based technique that makes use of accumulated "workload statistics" to accurately predict the "stall cycle counts" caused due to bus contention. This eliminates the need to simulate arbitration on every bus access, resulting in substantial speed-up. It is assumed that each Processor in the system has a distinct fixed priority, and arbitration is based on priority. We verify accuracy of our proposed model against results achieved by cycle accurate simulation. Two kinds of traffic is used for experiments. Synthetically generated traffic as well as traffic from real-world application is used to verify the bus model. We report an accuracy with an error range of 0.1% - 5% for the synthetic traffic as well as achieving a speedup of 7x on average. For the real traffic, we use a limited "single blocking" bus model and report results accordingly. (*Abstract*)**

Keywords— bus; statistical model; performance prediction; arbitration.

I. INTRODUCTION

System-On-Chips are rapidly moving towards multiprocessor architectures (MPSoCs). With the increasing complexity of the MPSoCs and on-chip communication architecture under short time-to-market, there is a growing need for faster and accurate design techniques. There is also a growing demand for a highly optimized design at the system-level, on MPSoC architecture as well as application software.

The performance and efficiency of MPSoCs depends heavily upon the efficiency of it's interconnect. Therefore, there is a need for better and faster techniques for analysis and optimization of on-chip interconnect. Currently, a lot of MPSoCs use bus based communication architectures such as Core-Connect from IBM, AMBA from ARM, SiliconBackplane from sonics etc[1]. Whereas MPSoCs with many more Processing Blocks are employing Networks-on-chip (NoCs) as the communication infrastructure.

In this paper we present a novel performance estimation technique for on-chip bus based architecture. Our approach includes using a statistical based model to accurately predict the stalls caused due to bus contention for a given application. The rest of the paper is organized as follows: section II presents related works. Section III gives a comparison of the simulation model vs the statistical based model. Section IV explains some key concepts that will be used in the rest of the document. Section V and Section VI, being the central parts of the document; explain a simple and a more complex mathematical bus model, respectively. Section VIII shows the experiments and results. Section IX gives a brief conclusion.

II. RELATED WORKS

In this section, we review existing performance evaluation approaches of on-chip communication architectures. Some approaches focus on simulation based performance estimation. As shown in a previous work in [2]. Simulation approaches usually tend to be more accurate but are very costly on simulation time. Commercial tools employ simulation at various abstraction levels [3]. Work in [4] uses static performance estimation techniques. Static techniques are not able to capture the dynamic nature of bus contention. [5] Employs semi Markov process model for estimating performance of multiple-bus communication architecture. Work done in [6] uses generalized semi Markov process (GSMP) model and report an accuracy of 84% compared to simulation. Our proposed approach is different to previous works in that (a) compared to simulation methods its significantly faster (b) our approach uses actual recorded workloads of an application as input hence enabling the model to capture actual dynamic behavior of every application under study which results in the calculation being tailored to every application under study. As opposed to previous approaches, our model does not assume any specific traffic pattern and can be used for several different kinds of applications and SoC configurations.

III. COMPARING SIMULATION MODEL WITH STATISTICAL MODEL

In a simulation based approach, on every bus access, a global scheduler dispatches a processor on processor queue to the simulator where the processor queue is sorted by processor's simulation clock, or in the case of a tie, by processor's priority on the bus access. Multiple bus requests are arbitrated by cycle-accurate bus simulation that leads to a huge computational load on the simulation. As shown in figure 1 the statistical based approach works such that for a predefined window of "T" cycles of simulation, arbitration delay on the bus is ignored. During these T cycles, histograms on all bus workloadsh_{Bi} and computation workloads $h_{Li}$are collected. These statistics are used by a mathematical model to calculate the expected arbitration delay per request$E[D_i]$. The total expected delay,$E[D_i] \times N_i$ is added to the total number of cycles.

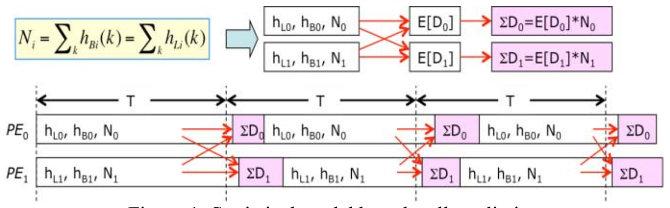


Figure 1: Statistical model based stall prediction

IV. KEY CONCEPTS

A. Computation Workload

Computation workload L_i denotes the number of execution cycles between two successive bus accesses on a Processing Element PE_i where i indicates the priority of a PE. Let N_i be the total number of computation workloads on PE_i.

B. Bus Workload

Bus workload B_i denotes the number of execution cycles between two successive computation workloads on PE_i. Total number of bus workloads onPE_i are same as N_i.

C. Request Probability and average interval workload

Let λ_i be the probability that a bus request r_i occurs at each cycle on PE_i. Probability that interval L_i equals n (cycles) is given as:

$$\Pr(L_i = n) = \lambda_i(1-\lambda_i)^{n-1} \ \ (n \geq 1)$$

Here, expectation of interval L_i ($E[L_i]$) is given as:

$$E[L_i] = \sum_{n=1}^{\infty} \Pr(L_i = n) \cdot n = \frac{1}{\lambda_i} \qquad (1)$$

Hence request probability is given as:

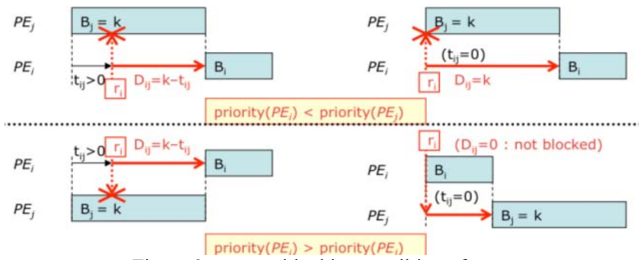
$$\lambda_i = \frac{1}{E[L_i]} \qquad (2)$$

D. Request inactivation Probability

On each occurrence of bus workload B_j, probability that request r_i *does not occur* within the duration of $B_j - 1$ cycles on PE_i is called r*equest inactivation probability* Y'_{ij}.

V. MATHEMATICAL MODEL(SINGLE BLOCKING MODEL)

from here on we will use following terminologies throughout this document, r_i : a request event by processor PE_i, b_j : a bus event (with arbitrary length B_j) at processor PE_j, blk_{ij} : a *blocking* event by bus event b_j on request r_i , $b_j(k)$: a bus event with length $B_j = k$ at processor PE_j, $blk_{ij}(k)$: a *blocking* event by bus event $b_j(k)$ on request r_i, $t_{ij}(k)$: time difference between the first cycle of $b_j(k)$ and request r_i.
$E[D_{ij}]$: Expectation of bus stall per request at processor PE_i.
$E[D'_{ij}]$: Expectation of bus stall per request at processor PE_j.
First we introduce a simple bus model such that, (a) bus requests on all PE are generated randomly with probabilityλ_i (b) Blocked requests must be granted after the current workload on the bus finishes.



PE_j
PE_i
$B_j = k$
$t_{ij}>0$
r_i
$D_{ij}=k-t_{ij}$
B_i
$(t_{ij}=0)$
$D_{ij}=k$
priority(PE_i) < priority(PE_j)
$(D_{ij}=0$: not blocked)
priority(PE_i) > priority(PE_j)

Figure 2: request blocking condition of request r_i

The Binary value α_{ij} describes the *fixed* priority relationship Between processors PE_i and PE_j.

$$\alpha_{ij} = \begin{cases} 0 & \left(priority(PE_i) < priority(PE_j)\right) \\ 1 & \left(priority(PE_i) > priority(PE_j)\right) \end{cases} \qquad (3)$$

As shown in Figure 2 request blocking condition of request r_i by bus event $b_j(k)$ is given as follows:

$$\alpha_{ij} \leq t_{ij}(k) \leq k-1 \qquad (4)$$

A. Bus stall expectation equation

First, bus event b_j that potentially blocks the request r_i is further categorized into two subclasses namely, (a) Consecutive bus event bb_{ij}: bus event b_j at processor PE_j follows immediately after (with no interval) a bus event b_i at processor PE_i. Let S_{ij} be the probability of event bb_{ij} i.e. $S_{ij} = \Pr(bb_{ij})$. (b)Non-consecutive bus event$\overline{bb_{ij}}$: bus event b_j at processor PE_j follows a bus event b_i at processor PE_i after one or more cycles. Clearly:$\overline{S_{ij}} = \Pr(\overline{bb_{ij}}) = 1 - \Pr(bb_{ij}) = 1 - S_{ij}$.Here, among N_j occurrences of event b_j on processorPE_j, there are $N_j \cdot S_{ij}$ occurrences of event bb_{ij} and $N_j \cdot (1 - S_{ij})$ occurrences of event $\overline{bb_{ij}}$.
Furthermore, events bb_{ij} and $\overline{bb_{ij}}$ are annotated as $bb_{ij}(k)$ and $\overline{bb_{ij}}(k)$ to denote the specific bus workload length k of $b_j(k)$. Also, let $S_{ij}(k) = \Pr\left(bb_{ij}(k)\right)$ and $\overline{S_{ij}}(k) = \Pr\left(\overline{bb_{ij}}(k)\right)$ be the probabilities of event $bb_{ij}(k)$ and $\overline{bb_{ij}}(k)$. If we assume that bus events $b_j(k)$ with different lengths k are generated randomly with probability $f_{Bj}(k)$, then $S_{ij}(k)$ and $\overline{S_{ij}}(k)$ are given as

$$\begin{cases} S_{ij}(k) & = & S_{ij} \cdot f_{Bj}(k) \\ \overline{S_{ij}}(k) & = & \overline{S_{ij}} \cdot f_{Bj}(k) = (1 - S_{ij}) \cdot f_{Bj}(k) \end{cases}$$

Figure 3 illustrates the blocking probabilities and bus stall delays on event $bb_{ij}(k)$ with specific values of $t_{ij}(k)$. Here, the probability of $t_{ij}(k) = n$ on event $bb_{ij}(k)$ is given as

$$\Pr\left(t_{ij}(k) = n | bb_{ij}(k)\right) = \lambda_i \cdot (1-\lambda_i)^{n-1} \qquad (5)$$

Whereas bus stall length is given as:

$$D'_{ij}\left(k | t_{ij}(k) = n, bb_{ij}(k)\right) = k - n \qquad (6)$$

Then the conditional bus stall delay on event $bb_{ij}(k)$ are:

$$E[D'_{ij}(k) | bb_{ij}(k)] = \frac{k\lambda_i - 1 + (1-\lambda_i)^k}{\lambda_i} \qquad (7)$$

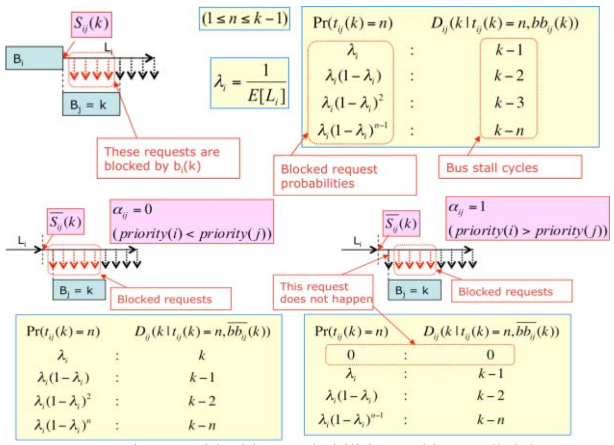

$S_{ij}(k)$
B_i
L_i
$B_j = k$
These requests are blocked by $b_i(k)$
$(1 \le n \le k-1)$
$\lambda_i = \frac{1}{E[L_i]}$
$\Pr(t_{ij}(k)=n)$
$D_{ij}(k \mid t_{ij}(k)=n, bb_{ij}(k))$
λ_i : $k-1$
$\lambda_i(1-\lambda_i)$: $k-2$
$\lambda_i(1-\lambda_i)^2$: $k-3$
$\lambda_i(1-\lambda_i)^{n-1}$: $k-n$
Blocked request probabilities
Bus stall cycles
$\overline{S_{ij}}(k)$
$\alpha_{ij}=0$ $(priority(i) < priority(j))$
L_i
$B_j = k$
Blocked requests
$\overline{S_{ij}}(k)$
$\alpha_{ij}=1$ $(priority(i) > priority(j))$
L_i
This request does not happen
$B_j = k$
Blocked requests
$\Pr(t_{ij}(k)=n)$
$D_{ij}(k \mid t_{ij}(k)=n, \overline{bb_{ij}}(k))$
λ_i : k
$\lambda_i(1-\lambda_i)$: $k-1$
$\lambda_i(1-\lambda_i)^2$: $k-2$
$\lambda_i(1-\lambda_i)^n$: $k-n$
$\Pr(t_{ij}(k)=n)$
$D_{ij}(k \mid t_{ij}(k)=n, \overline{bb_{ij}}(k))$
0 : 0
λ_i : $k-1$
$\lambda_i(1-\lambda_i)$: $k-2$
$\lambda_i(1-\lambda_i)^{n-1}$: $k-n$
Figure 3: blocking probabilities and bus stall delays

Figure 3 illustrates the blocking probabilities and bus stall delays on event $\overline{bb_{ij}}(k)$ with specific values of $t_{ij}(k)$. The probability of $t_{ij}(k) = n$ and its corresponding bus stall delay are:

$$\Pr\left(t_{ij}(k) = n | \overline{bb_{ij}}(k)\right) = \begin{cases} \lambda_i(1-\lambda_i)^n & (\alpha_{ij} = 0) \\ \lambda_i(1-\lambda_i)^{n-1} & (\alpha_{ij} = 1) \end{cases} \quad (8)$$

$$D'_{ij}\left(k | t_{ij}(k) = n, \overline{bb_{ij}}(k)\right) = k - n \quad (10)$$

Where the effective ranges of n on the above expressions are $\alpha_{ij} \le n \le k-1$. Then the conditional bus stall delay on event $\overline{bb_{ij}}(k)$ are:

- $\alpha_{ij} = 0$:

$$E\left[D'_{ij}(k) | \overline{bb_{ij}}(k)\right] = \frac{(k+1)\lambda_i - 1 + (1-\lambda_i)^{k+1}}{\lambda_i} \quad (9)$$

- $\alpha_{ij} = 1$:

$$E\left[D'_{ij}(k) | \overline{bb_{ij}}(k)\right] = \frac{k\lambda_i - 1 + (1-\lambda_i)^{k}}{\lambda_i} \quad (10)$$

Next, the overall bus stall delay expectation $E\left[D_{ij}(k)\right]$ on all bus events $bb_{ij}(k)$ are derived as follows:

$$E\left[D'_{ij}(k)\right] = E\left[D'_{ij}(k) | bb_{ij}(k)\right] \Pr\left(bb_{ij}(k)\right) + E\left[D'_{ij}(k) | \overline{bb_{ij}}(k)\right] \Pr\left(\overline{bb_{ij}}(k)\right) \quad (11)$$

$$E\left[D_{ij}(k)\right] = Q_{ij} E\left[D'_{ij}(k)\right] \quad (12)$$

Where $Q'_{ij} = \frac{N_j}{N_i}$. Here $D'_{ij}(k)$ is the probability that $bb_{ij}(k)$ occurs among N_j bus events on processor PE_j. On the other hand, $D_{ij}(k)$ is the probability that $bb_{ij}(k)$ occurs among N_i bus events on processorPE_i. Since $R_{ij}(k)N_i = R'_{ij}(k)N_j$, the factor Q'_{ij} is introduced to convert the two probabilities observed on processors PE_j and PE_i.

- Over all $E\left[D_{ij}(k)\right]$ for $\alpha_{ij} = 0$ becomes:

From (11)

$$E\left[D_{ij}(k)\right] = Q'_{ij}\left(\frac{k\lambda_i - 1 + (1-\lambda_i)^k}{\lambda_i} \cdot S_{ij}(k) + \frac{(k+1)\lambda_i - 1 + (1-\lambda_i)^{k+1}}{\lambda_i} \cdot \overline{S_{ij}}(k)\right)$$

$$= \frac{Q'_{ij} f_{Bj}(k)}{\lambda_i}\left(k\lambda_i - U_{ij}\left(1 - (1-\lambda_i)^k\right)\right)$$

Where $U_{ij} = S_{ij} + (1-\lambda_i)(1-S_{ij})$

- Over $E\left[D_{ij}(k)\right]$ for $\alpha_{ij} = 1$ becomes:

$$E\left[D_{ij}(k)\right] = Q'_{ij}\left(\frac{k\lambda_i - 1 + (1-\lambda_i)^k}{\lambda_i} \cdot S_{ij}(k) + \frac{k\lambda_i - 1 + (1-\lambda_i)^k}{\lambda_i} \cdot \overline{S_{ij}}(k)\right)$$

$$= \frac{Q'_{ij} f_{Bj}(k)}{\lambda_i}\left(k\lambda_i - \left(1 - (1-\lambda_i)^k\right)\right)$$

Finally, the overall "bus stall" expectation,$E\left[D_{ij}\right]$, on all bus events b_j (with arbitrary length) are:

$$E\left[D_{ij}\right] = \sum_k E\left[D_{ij}(k)\right]$$

Therefore,

$$E\left[D_{ij}\right] = \begin{cases} Q'_{ij}\left(E\left[B_j\right] - U_{ij}\frac{\left(1 - V'_{ij}\right)}{\lambda_i}\right) & (\alpha_{ij} = 0) \\ Q'_{ij}\left(E\left[B_j\right] - \frac{\left(1 - V'_{ij}\right)}{\lambda_i}\right) & (\alpha_{ij} = 1) \end{cases} \quad (12)$$

Where, $Y'_{ij} = \sum_k f_{Bj}(k)(1-\lambda_i)^{k-1}$ and $V'_{ij} = (1-\lambda_i)Y_{ij}$

B. Consecutive bus-event probability

As shown in Figure 4 Let $bb_{ij}(k)$ be an event where a bus event b_j (with arbitrary length) immediately follows a bus event $b_i(k)$. This event $bb_{ij}(k)$ occurs on the following two cases, (a) When $blk_{ji}(k)$ occurs (bus event $b_i(k)$ blocks a request r_j) (b) When request r_j occurs immediately after bus event $b_i(k)$. Let $S_{ij} = \Pr\left(bb_{ij}\right)$. Due to space constraints the derivation is omitted, however S_{ij} is given as,

$$S_{ij} = \begin{cases} Q'_{ji}\left(1 - U_{ji}V'_{ji}\right) & (\alpha_{ji} = 0) \\ Q'_{ji}\left(1 - V'_{ji}\right) & (\alpha_{ji} = 1) \end{cases} \quad (13)$$

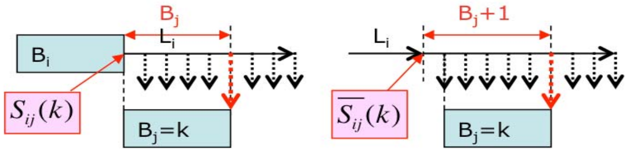


Figure 4: consecutive bus-event probability

C. Expressions for calculating bus stall

Using values of S_{ij} and λ_i from equations 13 and 2 respectively equation 11 becomes,

$$E\left[D_{ij}\right]\begin{cases}\alpha_{ji} = 0 \\ \alpha_{ji} = 1\end{cases} =$$

$$\begin{cases} Q'_{ij}\left(E\left[B_j\right] - \left(E\left[L_i\right] - 1\right)\left(1 - V'_{ij}\right)\right) - \left(1 - V'_{ji}\right)\left(1 - V'_{ij}\right) \\ Q'_{ij}\left(E\left[B_j\right] - E\left[L_i\right]\left(1 - V'_{ij}\right)\right) \end{cases}$$

Let us now discuss the bus event count ratio $Q'_{ij} = \frac{N_j}{N_i}$. Although N_i and N_j are the actual bus event counts observed during the bus predication interval, we need to take into account the fact that these bus event counts resulted while we ignored the bus stall delays, and therefore this will lead to inaccuracy if used directly. In order to include the bus stall delay effects in the bus event count ratio, we define the *average bus access interval* G_i as

$$G_i = E[L_i] + E[B_i] + E[D_i] \quad (16)$$

Where $E[L_i]$ is the interval workload expectation, $E[B_i]$ is the bus workload expectation, and $E[D_i]$ is the bus stall delay expectation:

$$E[D_i] = \sum_{i \neq j} E[D_{ij}] \quad (17)$$

Then the bus event count ratio Q_{ij} is calculated as

$$Q'_{ij} = \frac{E[L_i] + E[B_i] + E[D_i]}{E[L_j] + E[B_j] + E[D_j]} \quad (18)$$

Here, $E[D_i]$ and $E[D_j]$ are the predicted bus stall delays that will be calculated by iterative method.

VI. MATHEMATICAL MODEL(MULTI BLOCKING MODEL)

Now we introduce a more complex bus model such that, requests may be blocked by an infinite number of bus workloads. This model can be applied to "N" number of PEs such that *zero interval probability*$\mu_i > 0$. For this model we define two kinds of bus-workload. (a) "Raw bus workload" refers to the actual bus workload on each PE, as opposed to (b) "effective bus workload" which refers to a merged bus workloads. *Merged bus workload as* observed on PE_j by PE_i refers to the following situations (a)After termination of a bus workload on PE_j, a request is immediately generated with probability μ_j on PE_j such that $\alpha_{ij} = 0$. (b)After termination of a bus workload onPE_j, a bus workload immediately starts on PE_l with probabilityC_{jl}such that$\alpha_{il} = 0$. Then the probability of $L_i = n$ is:

$$\Pr(L_i = n) = \begin{cases} \mu_i & (n = 0) \\ (1 - \mu_i)\lambda_i(1 - \lambda_i)^{n-1} & (n \geq 1) \end{cases} \quad (19)$$

And, expectation of interval L_i ($E[L_i]$) is given as:

$$E[L_i] = \sum_{n=0}^{\infty} \Pr(L_i = n) \cdot n = \frac{1-\mu_i}{\lambda_i} \therefore \lambda_i = \frac{1-\mu_i}{E[L_i]} \quad (20)$$

In the above mentioned two cases, as shown in Figure5, a bus request r_i on the processor with lower priority sees a *merged* bus workload that potentially blocks its request. Note that merging can in fact occur consecutively as well.

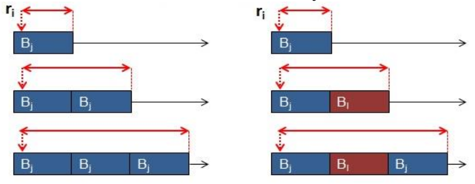


Figure 5: Merged bus workload

A. Calculation of merged busworkload

Bus request r_i sees a merged bus workload on higher priority PEs. Let's define, "C_{jl}": Probability that b_l immediately follows b_j (observed at PE_j). "$1 - \left(\sum_{l<i} C_{jl}\right)$": Probability that b_j is not followed by b_j or a bus workload on a higher priority PE compared to PE_i. The probability mass function for raw bus-workload $f_{Bj}(k)$ is divided as: $\begin{cases} 1 - \left(\sum_{l<i} C_{jl}\right) f_{Bj}(k) \\ C_{jl} f_{Bj}(k) \end{cases}$

Here, note that C_{jl} is defined against N_j bus-workloads on PE_j as opposed to S_{jl} that is defined against N_l bus-workloads on PE_l.

Let $E[B_{ij}]$ be expectation of *effective* bus workload B_{ij}. Let $E[B^*_{jl}]$ be expectation of *all merged bus-workloads* starting with b_j and terminating with b_l. Due to space constraints we omit the derivation. However , for "n" number of higher priority PEs, $\begin{bmatrix} E[B_{00}] & \cdots & E[B_{n0}] \\ \vdots & \ddots & \vdots \\ E[B_{0n}] & \cdots & E[B_{nn}] \end{bmatrix}$

$$= \begin{bmatrix} E[B_0] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E[B_n] \end{bmatrix} (I - C_n)^{-1} \quad (25)$$

Finally, the expectation of *effective* bus workload B_{ij} is,

$$E[B_{ij}] = \sum_{l<i} E[B^*_{jl}] \quad (26)$$

B. Calculation of Request inactivation probability

Here we show the expressions for calculating the effective *request inactivation probability*. Let Y^*_{ijk}be the effective *request inactivation probability* on PE_i over a merged bus-workload starting on $PE_j$and ending onPE_k. For "n" number of higher priority PEs,

$$\begin{bmatrix} Y^*_{i00} & \cdots & Y^*_{in0} \\ \vdots & \ddots & \vdots \\ Y^*_{i0n} & \cdots & Y^*_{inn} \end{bmatrix} = H_n(I - V_n C_n)^{-1} Y'_n \quad (26)$$

Where$V_n = \begin{bmatrix} V'_{i0} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & V'_{in} \end{bmatrix}$,$Y'_n = \begin{bmatrix} Y'_{i0} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Y'_{in} \end{bmatrix}$

Finally, the effective request inactivation probability is,

$$Y_{ij} = \sum_{l<i} Y^*_{ijl} \quad (27)$$

C. Expressions for calculating bus stall

The overall bus stall is given by using equation (12) with some differences.

$$E[D_{ij}] = \begin{cases} Q_{ij}\left(E[B_{ij}] - (w_{ij} + u_{ij})\dfrac{(1 - V_{ij})}{\lambda_i}\right) & (\alpha_{ij} = 0) \\ Q_{ij}\left(E[B_{ij}] - \dfrac{(1 - V_{ij})}{\lambda_i}\right) & (\alpha_{ij} = 1) \end{cases} \quad (12)$$

Notice that the effective bus-workload, effective request inactivation probability and effective event count ratio values are used. Also, instead of using the actual event count ratio we will use an effective bus event count such that $Q_{ij} = Q'_{ij} K_{ij}$ and

$$K_{ij} = \begin{cases} \left(1 - \mu_j - \sum_{l<i} S_{lj}\right) & (\alpha_{ij} = 0) \\ 1 & (\alpha_{ij} = 1) \end{cases}$$

$$w_{ij} = \begin{cases} (1 - \mu\_i)\, S\_ij \\ S\_ij \end{cases} \quad u_{ij} = \begin{cases} (1 - \lambda_i)(1 - S_{ij}) & (i > j) \\ 1 - S_{ij} & (i < j) \end{cases}$$

Some of the derivations in the mathematical model have been omitted due to space constraints.

VII. EXPERIMENTS AND RESULTS

Below we show how the model presented in our work will be used for performance prediction.

A. Input

The mathematical model uses "traffic workload statistics" as an input for its calculations. Histograms are maintained for "bus workload" and "computational workload". While, an "SoC configuration file", contains information about (a) Number of Processing Elements (b) Priority of every Processing Element (c) Number of buses (d) bus mapping (e) Memory. The bus mapping determines potential bus contention areas. Whereas the histograms are used by the mathematical model to predict the stall resulting from those contentions.

B. Traffic

For our experiment we use two kinds of traffic.

1) Syntheticaly generated traffic

We use a synthetic traffic generator that generates traffic with certain traffic characteristics as specified in a text file and can be changed easily by hand. The traffic characteristics include (a) Total number of traffic generators (PE) (b) ID of generator (c) Packet Length of generated traffic on each generator (d) Average interval between two successive packets on each generator and (e) total simulation time.

2) Recorded traffic patterns of real applications

We use a real-world JPEG encoder application to evaluate our prediction model in a real-world scenario. The reference program is made available by the Independent JPEG Group [7].At the moment due to technical issues we are able to report results based on this application only for our single blocking model.

C. SoC configuration

For our synthetic traffic experiments we use three different SoC configurations with 2, 3, and 4 PEs respectively. Figure7 shows 3 SoC configurations. All configurations have a single shared bus that connects the PEs. Whereas for real application we use a 8-PE system.

D. Simulation flow

The simulation flow consists of three phases. (i) In the Input phase, histograms of traffic statistics are read by the simulation engine. (ii) In the prediction phase, for a window of T cycles (bus prediction interval), "traffic statistics" are used by the prediction model to predict total bus stall cycle count for the prediction interval and added to the processor's simulation clock. (iii) Phase ii is repeated until the end of the traffic cycles is reached.

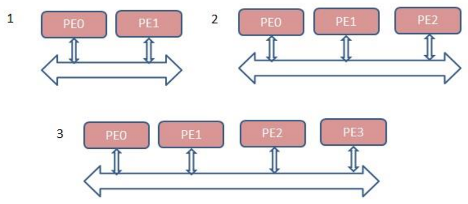


Figure 6: SoC configurations

E. Busstall calculated by simulation

For verification we compare our predicted stall cycle count with simulated stall cycle count. A comparison of simulation and prediction approach is reported in section III.

F. results

The results from our mathematical model are compared against results from simulation for verification. This simulation model has been a part of a previous work in [2].

1) result1

first we performed experiements for "SoCconfig1", that comprises of two PEs, using synthetically generated traffic. Table1 shows details of three different synthetically generated traffic on each PE such that PE(BW,CW) specifies the "ID", "average Bus-workload" and "average Computation – workload" on each PE. "sim.Stall" reports the "bus stall per request" calculated by simulation while "prd.Stall" reports the "bus stall per request" predicted using the proposed prediction model. "error" reports the error in predicted values calculated by the formula "(sim.Stall- prd.Stall)/ sim.Stall". Figure7 shows a comparison of "sim.Stall" and "prd.Stall" for three different traffic patterns and "percent error" in prediction vs simulation. The error in this case is under "0.1%".

Table1: Prediction results for 2PE system

PE(BW,CW)	sim.Stall	prd.Stall	error
P0(6,5)	8.67556	8.67436	0.00013832
P1(20,9)	3.864428	3.864936	-0.000131455
P0(20,5)	11.476614	11.47577	7.35409E-05
P1(20,9)	17.998777	17.99827	2.81686E-05
P0(7,25)	0.672208	0.671632	0.000856878
P1(7,25)	1.638099	1.636371	0.001054881

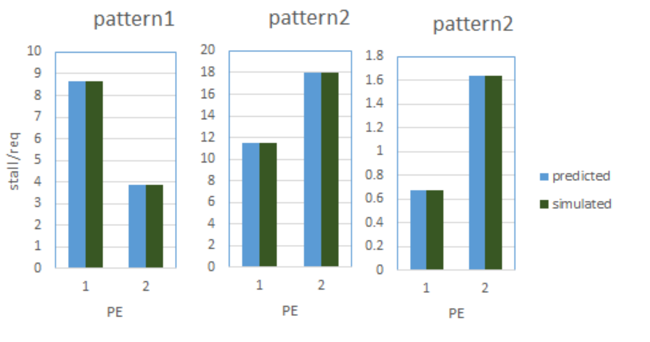


Figure 7: Comparison between predicted and simulated stall for 2PE system

2) result2

Experiment2 uses "SoC config2" which comprises of 3PEs. Table2 shows details of three different synthetically generated traffic on each PE. Figure8 shows a comparison of "sim.Stall" and “prd.Stall”. The prediction results are very accurate with a maximum error of around 3.3% as per given traffic patterns.

3) result3

In the next experiment we verify our model using “SoCconfig3” which comprises of 4PEs. Table3 shows details of the synthetically generated traffic, Figure9 compares “sim.Stall” and “prd.Stall”. The prediction results show an error of about 5.2% for stall predicted for the lowest priority PE. We compared execution times of the “simulation method” and “prediction method”. We were able to achieve on average 7x speedup. Note that the speed up results are shown for a simulation run for 800,000 cycles. For a simulation of 5000,000 cycles, simulating the bus arbitration takes in order of 1000ms. Given that the designer needs to evaluate multiple mapping, architecture configurations and tweak application accordingly, the overall speedup could be significant. Moreover, bus contention simulation increases with increasing bus requests whereas the time required for proposed prediction technique is the product of “number of time-windows” and “prediction time for each window”.

4) result4

Next we use a real-world application. Due to technical issues we are only able to report results for the “single-blocking request model” which only caters for two PEs at a time. We apply this model to a system with 8PEs. The predicted bus stalls, even though containing 13%-50% error, does help us get an over-all parallel record of the timing. Figure10 compares parallel record achieved using bus simulation against prediction model.

VIII. CONCLUSION

In this paper we introduced a novel stall prediction technique for MPSoC based on a statistical model. We used workload statistics to predict bus stalls. The effectiveness of our method is confirmed on a set of experiments on synthetically generated traffic. In the future we want to build on this work and perform more experiments on real-world applications.

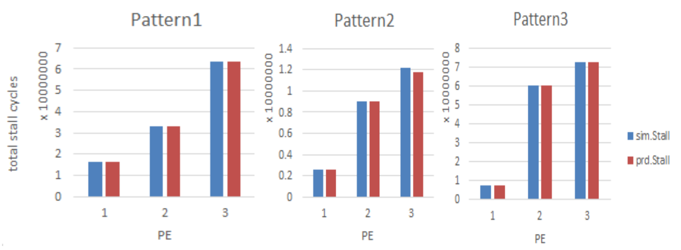


Figure 8: Comparison between predicted and simulated stall for 3PE system

Table 2: Prediction results for 3PE system

PE(BW,CW)	sim.Stall	prd.Stall	error
P0(7,5)	16305808	16296179	0.000591
P1(15,5)	32875215	32894991	-0.0006
P2(12,25)	63673825	63635272	0.000605
P0(5,25)	2550224	2554095	-0.00152
P1(5,25)	9000950	9014907	-0.00155
P2(5,25)	12207995	11798469	0.033546
P0(17,5)	7279891	7279375	7.09E-05
P1(5,5)	60163715	60161776	3.22E-05
P2(10,5)	72255797	72292485	-0.00051

Table 3: Prediction results for 4PE system

PE(BW,CW)	sim.Stall	prd.Stall	error
P0(5,14)	1.726473	1.725889	0.000338262
P1(5,14)	2.6	2.599	0.000384615
P2(5,14)	4.43577	4.58	-0.032515212
P3(5,14)	8.6	8.243	0.041511628
P0(8,18)	3.346597	3.346	0.00017839
P1(11,25)	3.9594	3.958	0.000353589
P2(5,14)	9.2566	8.925761	0.035740877
P3(9,29)	15.0364	15.831	-0.052845096

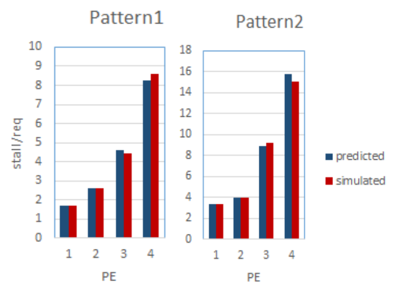


Figure 9: Comparison between predicted and simulated stall for 4PE system

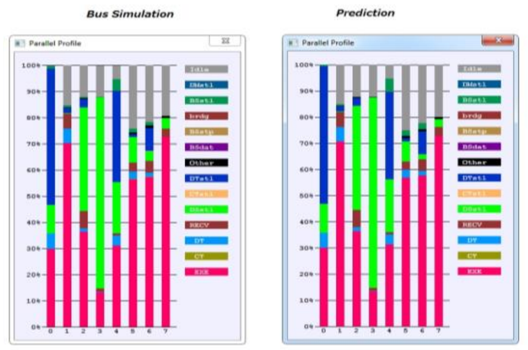


Figure 10: parallel record of “bus simulation” against “prediction model”

Table 4: Speedup

	Time required (simulation method) (msec)	Time required (Prediction method) (msec)	Speedup
Sim1	155	20	7.75x
Sim2	141	14	10x
Sim3	99	22	4.5x
Sim4	74	8	9.25x
Sim5	35	3	11.66
Avg	100.8	13.4	7.5x

REFERENCES

[1] www.ibm.com , www.arm.com , www.sonicsinc.com/.

[2] Tsuyoshi Isshiki , Li, Kunieda, Isomura, Satou, Trace-driven workload simulation method for Multiprocessor System-On-Chips, Proceedings of the 46th Annual Design Automation Conference, July 26-31, 2009, San Francisco, California.

[3] Synopsis CoCentric System Studio, CoWare N2C Design System.

[4] Daveau, Ismail, Jerraya, “Synthesis of system-level communication by an allocation-based approach,” in Proceedings of the Eighth International Symposium on System Synthesis, 1995,pp. 150–155

[5] Mudge, Al-Sadoun, “A semi Markov model for the perfor-mance of multiple-bus systems,” IEEE Transactions on Computers , vol.C-34, no. 10, pp. 768–783, Oct 1985.

[6] Ulhas Deshmukh, “Analytical Performance Estimation from GSMPModel for Hierarchical Bus-bridge Based SoCCommunication Architecture”, ICM 14-17 Dec. 2008. Sharjah.

[7] Independent Jpeg Group, www.ijg.org