ESS2013
2013/10/18

Regular Paper

MVA-based Probabilistic Model of Shared Memory with Fixed Priority Arbiter for Predicting Performance With Heterogeneous Workload

Ryo Kawahara,† Kouichi Ono† and Takeo Nakada†

Memory access contention can be a cause of performance problems and should be assessed at early stages of development. We devised a probabilistic model of shared memory for performance estimation. A fixed-priority arbiter with preemptive-repeat policy is modeled using Mean Value Analysis (MVA) based approximations. The calculation time is linear in the number of processors. The model is applicable for heterogeneous bandwidth utilization. Our model agrees well with a cycle-level Monte-Carlo simulation. Estimated execution time by our model is compared with the measured execution time of benchmark programs. We find a maximum error of 3.7% at a moderate bandwidth utilization, while a maximum error of 18.6% is found at a higher bandwidth utilization. This discrepancy can be explained by the periodicity of the access pattern.

並列処理による性能向上の阻害要因の 1 つに共有メモリーへのアクセスの競合があり，開発初期段階で評価できることが望ましい．本論文では preemptive-repeat 型の固定優先度調停の共有メモリーでの競合の確率モデルを提案する．本モデルはプロセッサー数に対し線形時間で計算でき、ヘテロジニアスな負荷でも評価可能である。サイクルレベルのモンテカルロシミュレーションとの比較は良好であり、ベンチマークを使った実験との精度の比較の結果，中程度の帯域利用率では最大で 3.7%、さらに高い領域では 18.6% の誤差となった。この誤差の原因はアクセスパターンの想定の違いで説明できる。

1. Introduction

Since embedded systems are increasingly large and complex, it is difficult to choose appropriate system architectures that satisfy the performance requirements. An embedded system usually has a heterogeneous architecture, which has many implementation choices, spanning hardware and software. To compete in today's markets, rapid development calls for assessing the system performance at early stages in the development process. A lightweight evaluation method is needed to prune large design spaces. Multi-processors or application-specific integrated circuits (ASICs) can be used to exploit parallelism to improve performance, but a shared memory can limit its effectiveness because of memory access contention. Thus the effects of memory access contention must be considered when estimating the performance in the system architecture design phase. Shared memory with a prioritized arbiter is especially of interest if we consider real-time applications.

There are various approaches to performance evaluation with various trade-offs between the evaluation speed, the accuracy, and the abstraction level of each model[1]. For example, the Queuing Network (QN) model is a widely used abstract model to evaluate resource contentions. However, this model cannot handle some behaviors intrinsic to software, such as synchronization. An ESL (Electronic System Level) simulation such as SystemC[2] can also be used to evaluate the memory access contention by describing all of the access timing, but the development of such fine-grained models is expensive.

One promising approach is a hierarchical modeling[3]. The behavior at the software or application level is described by appropriate models, such as task graphs, to calculate workloads for resources, while the resource contention at the lower level is calculated using other models (such as QN model). A similar approach[4] can be used with a Unified Modeling Language (UML) model, which is easier for software designers. In addition, the parameters necessary for calculating the workloads are obtained from abstracted execution traces[5] measured on an existing reference system. This means that the parameters can be obtained at low cost, since it is often the case that the system to be developed is based on the reference system.

In the hierarchical modeling approach, the designer needs to provide a model for estimating the degree of resource contention for each

† IBM Research - Tokyo

shared resource. This paper proposes a model of shared memory with fixed-priority arbiter for the same purpose. In Section 2, various methods are reviewed from this perspective. The method is described in detail in Section 3. We give experimental results in Section 4 and discuss those results in Section 5. Section 6 concludes the study.

2. Related work

Here we review some studies of the memory access contention from the viewpoints discussed in the previous section.

Hoogendoorn[6] and Mudge[7] studied the memory access contention using probabilistic models for an equal-priority arbiter. They incorporated the effects of re-submission of rejected accesses in their analysis to increase the accuracy. An application to hierarchical modeling was demonstrated by Kawahara et al.[8]. In their method, the model counts all the access patterns generated by N processors, which requires $O(2^N)$ calculations for a system with a heterogeneous workload distribution.

Smilauer[9] proposed a method based on Mean Value Analysis (MVA) for which the complexity of the calculations is polynomial in N. Sorin et al., also applied MVA to a queuing model of shared memory system with cache memories and bursty accesses[10] to show a good accuracy. However, their model requires at least 18 parameters for a processor. In those cases, the methods are still limited to the equal-priority arbitration.

Extensions to prioritized arbiters have been proposed by Yang et al.[11] and John et al.[12] for the discrete time case. In the latter study, the effect of re-submission is taken into account. However, these methods also have similar limitations on the complexity of calculation. For continuous time case, a prioritized queueing model can be used. Jaiswal obtained analytic results for various kinds of the priority queues. Unfortunately, he did not evaluate the delay of access by the queueing[13]. Application of MVA to preemptive-resume policy was proposed by Bryant et al.[14] but this queuing policy is not applicable for memory arbiters.

Another approach was proposed by Bobrek[15] et al. They proposed a method that exploits a non-linear regression model of the resource contentions. This kind of method can be applied to various kinds of shared memories[16] and also has an advantage in its simulation speed. However, such methods have high learning costs at modeling time.

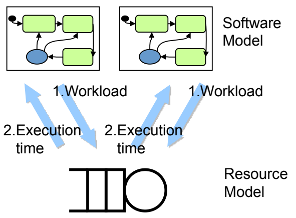


Fig. 1 A schematic description of a hierarchical modeling.

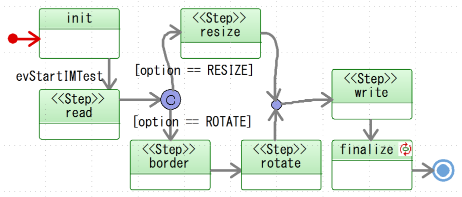


Fig. 2 Example of UML model (State chart) for performance simulation. A state with ⟨⟨Step⟩⟩ contains workload parameters.

We propose a probabilistic model of a shared memory with a fixed priority arbiter. The contributions of our work are as follows: (i) only two workload parameters are required for a processor, (ii) the contention of shared memory access can be calculated in a linear order of N, (iii) the method is applicable to both the short access time limit ($L = 1$ cycles) and the long access time limit ($L >> 1$). (iv) the method has better or at least competitive accuracy than the former theoretical results.

3. Proposed method

In this section, we describe our probabilistic model of a shared memory with fixed-priority arbiter. Before going into the detail, we first explain the overall simulation method for performance estimation of embedded systems and its connection to the probabilistic model. Symbols frequently appear throughout this paper is summarized in Table 1.

3.1 Hierarchical modeling

A hierarchical model consists of resource models and software models (Fig. 1). Our probabilistic model is a kind of the resource model for shared memory. For the software models, we use UML to describe it. An example of a UML model is shown in Fig. 2. A UML model is executed by a coarse-grained event-driven simulation. We use the word "coarse-

Table 1 List of symbols.

Symbols	Description
T_i	Step execution time of processor i.
U_i	Bandwidth utilization requested from processor i.
T'_i	Step execution time of processor i w/ contention.
U'_i	Bandwidth utilization requested from processor i w/ contention.
U_i^+	Sum of bandwidth utilization of processors with higher priorities than i.
N	Number of processors.
M	Number of memory accesses within a step.
L	Memory access time (#cycles for a memory to process an access w/o contention).
a_i	Average ratio of memory access time with contention.
x	Interval between two successive accesses normalized by L.
f_i	Distribution of x.
λ_i^+	Rate of accesses per cycle of an imaginary processor whose utilization is U_i^+.
c_i	Completion time of an access from processor i.
γ_i	Busy period of processor i.
$E(X)$	Average of X.

grained" to indicate that the system behavior is described with units that correspond to large blocks of code such as functions or tasks. and does not include instruction-level or code-level descriptions[8)][5)].

We call the basic unit a "step". Each step i has a workload parameter (T_i, U_i), where T_i is the step processing time and U_i is the bandwidth utilization requested when there is no memory access contention. The bandwidth utilization is defined as the ratio of the memory access throughput to its maximum determined by the hardware capacity. Assuming that a memory controller accepts either a read or a write access in one time, then the utilization can be calculated from the relation

$$U_i = \frac{M_{\mathrm{R},i}/T_i}{W_\mathrm{R}} + \frac{M_{\mathrm{W},i}/T_i}{W_\mathrm{W}}, \qquad (1)$$

where W_R and W_W are the maximum throughput of read and write memory accesses respectively, and $M_{\mathrm{R},i}$ and $M_{\mathrm{W},i}$ are the numbers of read memory accesses and of write memory accesses respectively. The numerators correspond to the actual throughput. Note that $M_{\mathrm{R},i}$, $M_{\mathrm{W},i}$ and T_i are usually measured from an existing system (the reference system) without contention.

The step processing time on the target system T'_i is determined by adding the contribution from the resource contention to T_i. The total execution time is the sum of the T'_is along the critical execution path. This paper focuses on describing a concrete calculation procedure for T'_i.

3.2 Overview of probabilistic model

In this method, we evaluate the memory access contention from the workload parameters without the information about the memory access timing. We use some approximations that are similar to those often made in theoretical analyses.[7)]

The first approximation is that the memory accesses occur uniformly and randomly over time within a step, unless contention arises. This is not true in general because of the burstiness[17)].

The second approximation is that, within a simulation step, the memory access timings between different processors are independent of each other. Again, this is a rough approximation, because there may be periodic accesses or synchronizations.

Despite these rough approximations, this method provides a practical way to prune the design space before entering into the detailed design.

The starting point of our probabilistic model is the following formula, which is similar to Amdahl's law[18)]. For processor i,

$$T'_i = T_i((1 - U_i) + a_i U_i), \ (U'_i = U_i T_i / T'_i), \qquad (2)$$

where $a_i = L'/L$ is the average memory access delay ratio and L' and L are the average memory access times with and without contention, respectively. From the definition in Eq. (1), the bandwidth utilization corresponds to the proportion of time in which a memory controller is processing memory accesses from processor i. The utilization with contention U'_i in Eq. (2) comes from the fact that the total number of memory access transactions should be the same with or without the contention $(U_i T_i = U'_i T'_i)$.

We call a simultaneous memory access a "col-

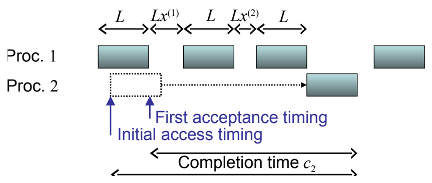


Fig. 3 Definition of the delayed access time L' and the completion time c. The case of two processors.

lision" in this paper. In the early studies[6)7)], a_i was estimated by counting all of the collision patterns. Let s_j be the state of processor j and $s = (s_1, \cdots, s_N)$. In the simple example of the access time L being one cycle, then $s_j = 1$ or 0, which correspond to the states of "accessing" or "not accessing", respectively. Then

$$a_i = \underbrace{\sum_{s_1}^{\{0,1\}} \cdots \sum_{s_N}^{\{0,1\}}}_{\text{except for } s_i} \frac{L'_i(s)}{L} P(s|s_i = 1), \qquad (3)$$

where $L'_i(s)$ is the delayed access time when the collision pattern is s, and $P(s|s_i = 1)$ is the conditional probability of the collision pattern being s when processor i is accessing memory. The delayed access time $L'_i(s)$ may include the length of a queue of memory access transactions, which depends on the arbitration scheme. We explicitly calculate a_i in the case of the fixed priority arbiter with some approximations.

3.3 Reduction of complexity

The sum over s in Eq. (3) requires $O(2^{N-1})$ calculations even in the simple case, which means it will be impossible to compute it when the number of processors N is large. To reduce the complexity of the calculations, we use an approximation in which we replace quantities such as the queue length in $L_i(s)$ with mean values of the quantities, as in the MVA approach by Smilauer[9)]. Then, the averaging over s can be done analytically without doing $O(2^N)$ calculations.

3.4 Fixed priority arbiter: $L >> 1$ case

There are several choices to implement a fixed-priority arbiter. Here, we assume the preemptive-repeat policy. When a higher-priority access preempts other access, the preempted access should be resubmitted.

The key ideas to derive the expression of a_i for the fixed-priority arbiter is that we introduce an imaginary processor whose bandwidth utilization is the sum of all the processors whose priority is higher than processor i, i.e.,

$$U^+_{i-1} = \sum_{k=1}^{i-1} U'_k, \quad (U'_k = U_k T_k / T'_k). \qquad (4)$$

Then the problem is reduced to the case of two processors. We regard the accesses from processors of higher priorities than processor i as accesses from the imaginary processor of which utilization is U^+_{i-1}. This is similar to the shadow approximation for preemptive-resume policy[14)], but we use this to adjust the service time instead of the service rate to handle "repeat" feature.

An example of arbitration between two processors is shown in Fig. 3. We first evaluate the completion time c_i, which is the time duration between the start of the acceptance of an access from processor i to the completion of the access[13)]. The completion time can be longer than L cycles because the process is repeated when there is a preemption of a higher priority access. We denote the average of c_i by $E(c_i)$.

If there are two successive memory accesses whose interval is less than L, then an access with lower priority cannot be completed during this interval. Let x be {the cycles of the interval between two successive accesses $/L$}. We consider the case of $L >> 1$ because of the long DRAM (dynamic random access memory) latency compared to the cycle time. Then, x can be considered as a continuous value. Using the assumption that the memory access is random, then x obeys to the exponential distribution.

$$f_{i-1}(x) = \lambda^+_{i-1} L \exp(-\lambda^+_{i-1} L x), \qquad (5)$$

where λ^+_{i-1} is the probability that the imaginary processor randomly generates an access transaction at each cycle if it is in the state of "not accessing".

The quantity $\lambda_i L$ is approximated as follows when $L >> 1$ and U_i kept constant.

$$\lambda^+_i L = \frac{U^+_i L}{(1 - U^+_i)L + 1} \approx \frac{U^+_i}{1 - U^+_i}. \qquad (6)$$

The denominator can be interpreted as the proportion of cycles where new generation of memory access is allowed in a duration. The average of x with a condition that the next access occurs within L cycles is

$$x_{i-1} = \int_0^1 dx x f_{i-1}(x)/Q_{L,i-1} \quad (7)$$

$$\approx \frac{(1-U^+_{i-1})}{U^+_{i-1}} - \frac{(1-Q_{L,i-1})}{Q_{L,i-1}}. \quad (8)$$

$Q_{L,i-1}$ is the probability that an access with higher priority occurs within L cycles after the previous access, i.e.,

$$Q_{L,i-1} = \int_0^1 dx f_{i-1}(x) \quad (9)$$

$$\approx 1 - \exp(-U^+_{i-1}/(1-U^+_{i-1})). \quad (10)$$

If there are n such successive accesses, then the completion time is

$$E(c_i) = L\left\{\sum_{n=0}^{\infty}(x_{i-1}+1)np(n) + 1\right\} \quad (11)$$

where

$$p(n) = (1-Q_{L,i-1})[Q_{L,i-1}]^n, \quad (12)$$

Using the formula of the sum of a geometric series, we obtain

$$E(c_i)/L = (x_{i-1}+1)\frac{Q_{i-1}}{1-Q_{i-1}} + 1. \quad (13)$$

If an access from processor i does not collide with an access with higher priority, then $E(c_i)$ is the delayed access time. Otherwise, the access from processor i should wait for the higher priority access to be done. This takes $L/2$ cycles in average if these accesses are independent. The probability of these two cases are $1 - U^+_{i-1}$ and U^+_{i-1} respectively. Thus the average memory access delay ratio becomes

$$a_i = (1-U^+_{i-1})\frac{E(c_i)}{L} \quad (14)$$

$$+U^+_{i-1}\left(\frac{1}{2} + \frac{E(c_i)}{L}\right) \quad (15)$$

$$= (x_{i-1}+1)\frac{Q_{i-1}}{1-Q_{i-1}} + \frac{1}{2}U^+_{i-1} + 1. \quad (16)$$

Here, we used the assumption of independence among processors (Section 3.2), thus processor i's states does not appear in these quantities.

As $U^+_{i-1} \to 1$, $Q_{L,i-1}$ also approaches one, and thus a_i diverges. This means that the bandwidth is occupied by higher priority accesses and the processor i never gets chance to access.

If we calculate a_i, T'_i and U'_i from $i = 1, 2, 3, \cdots$, then the complexity of the calculation is $O(N)$, because eqs.(14) depends only on the utilization of processors with higher priorities than processor i.

3.5 Fixed priority arbiter: $L = 1$ case

Although the case of the access time $L >> 1$ is important in practice, we derive the model in the case of the access time $L = 1$ to compare it with the previous study.

In this case, since memory accesses collide at discrete timings, we do not need to consider the interval between two successive accesses, i.e., $x_i = 0$ for all i. Similarly, $U^+_i = \lambda_i = Q_{L,i}$ holds. The model is simplified to

$$a_i = 1/(1-U^+_{i-1}). \quad (17)$$

3.6 Theoretical comparison with John et al.: $L = 1$ case

In their previous study[12], they calculated the acceptance ratio, which corresponds to $1/a_i$. Their result coincides with our result for the first two processors, and the difference appears in processors $i \geq 3$. For the case of three processors with the same utilization $U_i = U$, single shared memory and $L = 1$, their model reads

$$a_3^{-1} = a_2^{-1}(1 - U - U^2 + U^3). \quad (18)$$

3.7 Theoretical comparison with Jaiswal

The analytic form of the average completion time was obtained by Jaiswal for various queueing models[13]. Among those, preemptive-repeat-identical priority queueing model with the number of sources being one for each priority and the service time being a constant L corresponds to our case. His result coincides with our result for $i \leq 2$. For $i = 3$,

$$E(c_3) = \left[\frac{1}{\Lambda_2} + E(\gamma_2)\right]\left[\exp(\Lambda_2 L) - 1\right], \quad (19)$$

where

$$E(\gamma_2) = \frac{\lambda_2}{\Lambda_2}E(c_2) \quad (20)$$

$$+ \frac{\Lambda_1}{\Lambda_2}\left[L + E(c_2)(1 - \bar{S}_1(\lambda_1))\right], \quad (21)$$

is the average occupation period, which is similar to the busy period, and $\bar{S}_1(s) = \exp(-sL)$ is

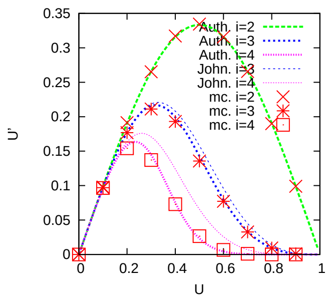


Fig. 4 Utilization assigned by the fixed-priority arbiter U' as a function of utilization request U.Estimates by our model (Auth.), John's model (John.) and results of Monte-Carlo (MC.) simulation are shown.

the Laplace transform of the service time distribution, and $\Lambda_i = \sum_{j=1}^{i} \lambda_j$ is the aggregate arrival rate.

The execution time T' was not evaluated in the study. We simply substitute Eq. (19) into Eq. (14).

4. Evaluation

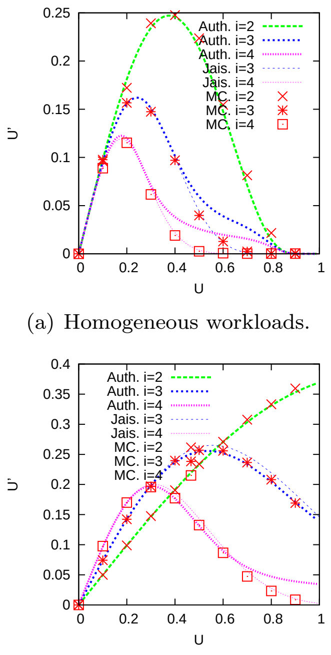
4.1 Comparison with cycle-level Monte-Carlo simulation

In this section, we compare the estimation by our method with the one by cycle-level Monte-Carlo (MC) simulation with the same conditions assumed in Section 3.2.

This MC simulation model consists of N processors and one shared memory with a fixed-priority arbiter. An i-th processor model has three states: accessing (AC), re-submission (RS), and not accessing (NA). The processor generates a memory access with a given probability if it is in the NA state, and changes its state according to the arbiter. In this MC simulation, $N = 4$ and the length of a simulation run for each plot is 10^4 cycles.

Estimates of the utilization after arbitration U' by our model, John's model and MC simulation are compared in Fig. 4 for $L = 1$ cycle. All the processors have same utilization requests $U_i = U$. Our model agrees better to the MC simulation.

Figures 5(a) and 5(b) are for the case of the access time $L = 20$ cycles. Utilization requests $U_i = U$ for all i are used in the case of the homogeneous workloads (Fig. 5(a)) and $U_1 = U/4, U_2 = U/2, U_3 = 3U/4, U_4 = U$ are used for the case of the heterogeneous workloads (Fig. 5(b)). Results of Jaiswal's model in



(a) Homogeneous workloads.

(b) Heterogeneous workloads.

Fig. 5 Utilization assigned by the fixed-priority arbiter U' as a function of utilization request U. Estimates by our model (Auth.), by Jaiswal's model (Jais.) and results of Monte-Carlo (MC.) simulation are shown.

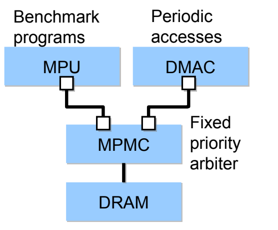


Fig. 6 Block diagram of experimental system on Xilinx ML510 FPGA board.

Section 3.7 are also shown. In the both cases, the models agree with the MC simulation qualitatively, but our model overestimates in the high utilization regions. Jaiswal's model overestimates in the medium utilization region in the case of the heterogeneous workloads.

4.2 Comparison with benchmark programs

We show our experimental results in this section. The experimental system consists of a microprocessor (MPU), a direct memory access (DMA) controller , multi-port memory controller (MPMC) and a shared memory, imple-

Table 2 System constants and measured parameters. Burst length (BL, in 32-bit words), read and write throughput (R TP and W TP, in 10^6 bursts/s) are shown.

(a) System constants.

	BL	R TP	W TP
MPU	8	19.68	16.67
DMA	16	13.33	11.77

(b) Parameters

pg	T (s)	$M_{\mathrm{R}}(\times 10^6)$	$M_{\mathrm{W}}(\times 10^6)$	U
jp	0.412	0.8226	0.3258	0.153
la	62.337	16.2544	5.0265	0.0181
ma	1.117	1.7621	0.9480	0.137
tb	1.921	5.5888	3.1883	0.260
tr	4.061	12.9010	8.5785	0.305
td	3.061	2.9707	2.1318	0.0967
tm	5.204	12.8343	6.5853	0.210
ty	4.169	12.0594	7.3973	0.267

mented on a Xilinx ML510 FPGA board[19) (Fig. 6). We used the Consumer program set from the MiBench benchmark suite[20) as the workload for the microprocessor, and periodic memory access patterns were generated by the DMA controller. Our intention for this combination of workloads was to reproduce the memory accesses in a system of software and ASICs working together. MPMC is configured to use a fixed-priority arbiter. We use the same hardware configuration throughout this experiment.

The Consumer set in MiBench is designed to represent consumer products such as video games and digital cameras, which are regarded as memory-intensive. It consists of eight programs: `jpeg` (jp), `lame` (la), `mad` (ma), `tiff2bw` (tb), `tiff2rgba` (tr), `tiffdither` (td), `tiffmedian` (tm), and `typeset` (ty).

First we obtained the workload parameters T and U of the benchmark programs in a single processor environment. Since each program has a single function, we regard one program as a simulation step and used averaged utilization for each step. The system constants and the obtained parameters are summarized in Tables 2(a) and 2(b), respectively.

When measuring the execution time, we used the `time` command. A constant time $t = 0.161$(s) (the time to execute an empty program) was subtracted from the measured time to reduce the effects of the execution overhead caused by the `time` command. The execution times in the tables are averages over 10 samples. The maximum standard deviation $\Delta T = \pm 0.019$ (s).

To measure the number of memory accesses from the microprocessor, we used a L1 cache-miss counter in an in-circuit cache emulator since any memory accesses to the shared memory are cache misses. The L2 caches were not used. The throughput constants were obtained from the specifications of the memory controller of the FPGA board. The utilization of the periodic accesses by the DMA controller was $U_{\mathrm{DMA}} = 0.180, 0.359$, and 0.539. This was adjusted by controlling the number of memory accesses within a period.

The results are summarized and compared with the simulation results in Figs 7. Figure 8 shows an example of the execution time.

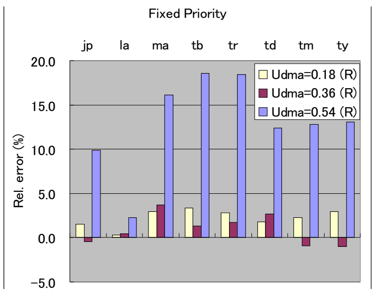


Fig. 7 Error in estimated execution times. Result of the proposed probabilistic model (Rand) is also shown.

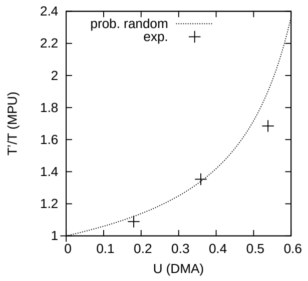


Fig. 8 Execution time of benchmark `typeset`. Estimations by our model (prob. random) is also shown.

5. Discussion

As we see in Section 4.1, Jaiswal's model agree better with the MC simulation in the high utilization region than our model. This is because our model ignores the correlations generated between the higher priority accesses as a result of the arbitration, i.e., they tend to be clustered since a preempted access is accepted just after the preempting access is completed. However, Jaiswal's model ignores the decay of the aggregate arrival rate Λ_i due to the contention. Our model handles this effect by incorporating the aggregate utilization U_i^+ after the arbitration. This is why our model agrees better in the case of the heterogeneous workloads.

In the comparison with the benchmark experiments in Section 4, we saw a relatively large error at $U_{\mathrm{DMA}} = 0.54$. The reason of this discrepancy seems to be in the periodicity of the DMA. Since the period is kept constant in the experiment, the period limits the length of the completion time. Therefore our model overestimates the result. Thus, we need to incorporate additional parameters, such as the period of the access pattern, to increase the accuracy at higher utilization range.

6. Conclusion

We devised probabilistic model of shared memory with fixed-priority arbiter applicable to performance estimation in early stages of development of embedded systems. The calculation time of the model is $O(N)$ due to the mean value approximations. The model is applicable to a shared memory of which access time is much longer than one cycle as well as one cycle case.

Our model is first compared to a cycle-level Monte-Carlo simulation and shows at least competitive accuracy to the former studies. Next, the estimated execution time with our model is compared with the measured execution time of benchmark programs from MiBench suite with memory access contention. We find a maximum error of 3.7 % at a moderate bandwidth utilization ($U \lesssim 0.4$), while a maximum error of 18.6 % is found at a higher bandwidth utilization ($U = 0.54$). This discrepancy can be explained by the periodicity of the access pattern.

Our future work will include the application of our method to real embedded systems with measurement and modeling methods. Currently, our method does not handle architecture change of cache memories, i.e., one needs to subtract the number of cache hits from the workload parameters of the probabilistic model. In future, we need to identify a performance model of cache memory which can be used in conjuction with our method. Integration of Jaiswal's model and our model would also be interesting.

Acknowledgment
The authors would like to thank Kenta Nakamura and Hiroki Nishiyama for fruitful discussions on the experiments.

References

1) Gries, M.: Methods for evaluating and covering the design space during early design development, *Integration, the VLSI Journal*, Vol. 38, No. 2, pp. 131–183 (2004).
2) Open SystemC Initiative (OSCI): *SystemC specification* (2007).
3) Jonkers, H., van Gemund, A. and Reijns, G.: A probabilistic approach to parallel system performance modelling, *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, Vol. 2, pp. 412 –421 (1995).
4) Cortellessa, V., Pierini, P. and Rossi, D.: Integrating Software Models and Platform Models for Performance Analysis, *IEEE Transactions on Software Engineering*, Vol. 33, No. 6, pp. 385–401 (2007).
5) Ono, K., Toyota, M., Kawahara, R., Sakamoto, Y., Nakada, T. and Fukuoka, N.: A Model-based Method for Evaluating Embedded System Performance by Abstraction of Execution Traces, *Proc. of 6th European Conference on Modelling Foundations and Applications (ECMFA 2010)*, Lecture Notes in Computer Science, Vol. 6138, Springer, pp. 233–244 (2010).
6) Hoogendoorn, C. H.: A General Model for Memory Interference in Multiprocessors, *IEEE Transactions on Computers*, Vol. C-26, No. 10, pp. 998–1005 (1977).
7) Mudge, T. N., Hayes, J. P., Buzzard, G. D. and Winsor, D. C.: Analysis of Multiple-Bus Interconnection Networks, *Journal of Parallel and Distributed Computing*, Vol. 3, pp. 328–343 (1986).
8) Kawahara, R., Nakamura, K., Ono, K., Nakada, T. and Sakamoto, Y.: Coarse-grained simulation method for performance evaluation a of shared memory system, *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pp. 413–418 (2011).
9) Smilauer, B.: General Model for Memory Interference in Multiprocessors and Mean Value Analysis, *IEEE Transactions on Computers*, Vol. C-34, pp. 744–751 (1985).
10) Sorin, D., Lemon, J., Eager, D. and Vernon, M.: Analytic evaluation of shared-memory architectures, *IEEE Transactions on Parallel and*

Distributed Systems, Vol. 14, No. 2, pp. 166–180 (2003).

11) Yang, Q. and Ravi, R.: Design and analysis of multiple-bus arbiters with different priority schemes, *International Conference on Databases, Parallel Architectures and Their Applications (PARBASE-90)*, pp. 238 –247 (1990).

12) John, L. and Liu, Y.-C.: Performance model for a prioritized multiple-bus multiprocessor system, *IEEE Transactions on Computers* , Vol. 45, No. 5, pp. 580 –588 (1996).

13) Jaiswal, N. K.: *Priority Queues*, Academic Press, chapter 3, pp. 52–82 (1968).

14) Bryant, R. M., Krzesinski, A. E., Lakshmi, M. S. and Chandy, K. M.: The MVA priority approximation, *ACM Trans. Comput. Syst.*, Vol. 2, No. 4, pp. 335–359 (1984).

15) Bobrek, A., Paul, J. M. and Thomas, D. E.: Stochastic Contention Level Simulation for Single-Chip Heterogeneous Multiprocessors, *IEEE Transactions on Computers*, Vol. 59, pp. 1402–1418 (2010).

16) Poe, J., Cho, C.-B. and Li, T.: Using Analytical Models to Efficiently Explore Hardware Transactional Memory and Multi-Core Co-Design, *20th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD '08)*, pp. 159 –166 (2008).

17) Darema-Rogers, F., Pfister, G. F. and So, K.: Memory access patterns of parallel scientific programs, *Proceedings of the 1987 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, SIGMETRICS '87, pp. 46–58 (1987).

18) Hennessy, J. L. and Patterson, D. A.: *Computer Architecture, Fourth Edition, A Quantitative Approach*, Elsevier, Morgan Kaufmann Publishers, chapter 1, pp. 1–62 (2007).

19) Xilinx Inc.: *Xilinx ML510 Documentation* (2011).

20) Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T. and Brown, R. B.: MiBench: A free, commercially representative embedded benchmark suite, *Proceedings of the IEEE 4th Annual Workshop on Workload Characterization (WWC-4)*, pp. 3–14 (2001).