This document is a preprint of: R. Wittig, M. Hasler, E. Matus et al., "Statistical Access Interval Prediction for Tightly Coupled Memory Systems," in Proceedings of IEEE Symposium on Low-Power and High-Speed Chips (COOLCHIPS 2019), Yokohama, Japan, Apr 2019. DOI:10.1007/978-3-030-27562-4_16

Statistical Access Interval Prediction for Tightly Coupled Memory Systems

Robert Wittig, Mattis Hasler, Emil Matus and Gerhard Fettweis (Fellow IEEE)
Vodafone Chair for Mobile Communications Systems
Technical University Dresden
Email: {robert.wittig, mattis.hasler, emil.matus, gerhard.fettweis}@tu-dresden.de

Abstract

Sharing memory in embedded systems presents a promising approach to increase the area utilization of these constraint platforms. However, sharing inevitably results in access conflicts, which diminish the overall system performance. As a counter measure, we propose Access Interval Prediction. We argue that most memory transaction of embedded processors can be reliably predicted in the time domain. Therefore, preallocation of shared resources can be used to avoid collisions in the memory system. Our statistical model shows an accuracy of over 90 percent, thus significantly reducing memory contention.

I. Motivation and Overview

Current technology nodes enable mobile communication engineers to integrate complete modems as system on chip solutions [1], [2]. Communication between multiple processing elements (PEs) can be done via FIFOs or shared memory. The latter has the advantage of reduced energy consumption (no data transfers needed) but is susceptible for access conflicts. Banking SRAM resources can mitigate this problem [3]–[5], but again increases area and energy consumption. Hence, we propose Access Interval Prediction (AIP). We show how statistical information can be used to pre-allocate shared resources of tightly coupled memory systems to avoid conflicts. For evaluation, we assume the system shown in Fig. 1. A processor acts as a high priority master (HPM). In this paper we use a Xtensa LX6, commonly found in embedded platforms. Instruction and data port are connected to different memory banks. Further, one of the banks is shared with another low priority master (LPM). Depending on the shared bank, AIP is conducted for the instruction or data port of the HPM. Furthermore, we assume that the LPM is accessing the memory every cycle (e.g. data movement engine). As software we use the integer testsuite of the wildly acknowledged SPEC CPU benchmark. The described system is simple enough to focus on the performance of AIP but can be extended for the use of multiple HPMs and memory banks.

II. Theory

We define $\boldsymbol{d} = \boldsymbol{d_i}$ as the i-th randomly distributed access interval between two consecutive memory transactions of a single PE. An observation of $\boldsymbol{d}$ can take realizations between 1 and L and is denoted as $d_i \in \{1, ..., L\}$. For every interval i we want to have an estimator $\hat{\boldsymbol{d}}$ such that the error $\tilde{\boldsymbol{d}} = \boldsymbol{d} - \hat{\boldsymbol{d}}$ becomes zero with high probability: $\max P(\tilde{d} = 0)$. The result, if no further observations are available, is a maximum likelihood (ML) estimator: $\hat{\boldsymbol{d}} = \arg\max_{\boldsymbol{d}} P(\boldsymbol{d})$.

If we have access to previous observations $\boldsymbol{y} = \boldsymbol{y_k} = [\boldsymbol{d_{i-k}}, ..., \boldsymbol{d_{i-1}}]$, we can use a maximum a posteriori (MAP) estimator: $\hat{\boldsymbol{d}}_{\boldsymbol{i}|\boldsymbol{y}} = \arg\max_{\boldsymbol{d}} P(\boldsymbol{d}|\boldsymbol{y})$. In order to calculate the MAP estimator, the joint probability mass functions (pmf) $f_{d|y}(d|y) = P(\boldsymbol{d} = d|\boldsymbol{y})$ are required.

The probability of a correct prediction is given by: $P(\hat{\boldsymbol{d}} = \boldsymbol{d})$. In this case we can pre-arbitrate the HPM port with no penalty. If the predicted interval is longer than the true interval $P(\hat{\boldsymbol{d}} > \boldsymbol{d})$ a conflict penalty is incurred, because LPM and HPM want to access the memory at the same time. If, in contrast, the predicted interval is shorter than the true interval $P(\hat{\boldsymbol{d}} < \boldsymbol{d})$, two different scenarios are possible. 1) We can block the bank exclusively for the HPM port (priority blocking). This entails no conflict, but reduces the throughput of the LPM. 2) The LPM can access the bank after the miss-prediction (free-for-all). In this case the next access of the HPM will result in a conflict but the throughput of the LPM is not diminished. Thus, the probability of a conflict free access is given by

$$P_{\text{cfa}} = \begin{cases} P(\hat{\boldsymbol{d}} = \boldsymbol{d}) & \text{free-for-all} \\ P(\hat{\boldsymbol{d}} = \boldsymbol{d}) + P(\hat{\boldsymbol{d}} < \boldsymbol{d}) & \text{priority blocking.} \end{cases} \tag{1}$$

Moreover, we introduce the *interval* utilization. It measures the normalized memory utilization of the LPM. The difference between the *absolute* and *interval* utilization is illustrated in Fig. 2

III. Evaluation and Implementation Considerations

Fig. 3 depicts the probability of a conflict free access for the instruction and data port. Further, we differentiate between *priority blocking* and *free-for-all* mode. For readability, we averaged P_{cfa} over all testcases. In free-for-all mode, the ML estimator ($k = 0$) performs better at the instruction port ($P_{\text{cfa}} = 71\,\%$) than at the data port ($P_{\text{cfa}} = 33\,\%$). In contrast, the MAP estimator ($k > 0$) has a deeper impact at the data port. For $k > 2$ the probability of a conflict free access is already greater than at the instruction port. Furthermore, we observed that: $\lim_{k\to\infty} P_{\text{cfa}} = 100\,\%$ at both ports. This was to be expected because for $k \to \infty$ every access can be identified by its unique history. However, it is noteworthy that prediction rates of over $90\,\%$ can already be achieved for $k > 4$. In comparison, architectures relying solely on an interleaved memory bank system [6] only achieve a probability of $P_{\text{cfa}} = 75\,\%$ in the given scenario[1]. With priority blocking, the ML estimator achieves $P_{\text{cfa}} = 100\,\%$ at the instruction port. The reason is that most intervals at the instruction port have an interval of one cycle, which is the prediction of the ML estimator. In consequence, the prediction can either be true or too short. In the latter case, the memory gets blocked for the HPM, which also results in a conflict free access (equation 1). This is also the reason why the MAP estimator performs slightly worse for increasing k. However, for $k \to \infty$, the probability approaches $100\,\%$ at both ports.

The interval utilization is the second key metric for the evaluation of AIP, which is shown for the blocking mode in Fig. 4. The ML estimator achieves $0\,\%$ at the instruction port. Again, this is due to the fact that every access is predicted with an interval of one cycle and the memory is blocked for false predictions. The performance of the MAP estimator improves considerably for $k > 3$, because more estimations have an interval $\hat{d} > 1$.

AIP can result in a speedup, if predictions are correct and subsequent conflict penalties are avoided. The speedup is dependent on the penalty C incurred by an access conflict. Fig. 6 shows the speedup for different values of C over P_{cfa}. For $P_{\text{cfa}} = 0$ every access would result in a conflict. In contrast, with $100\,\%$ accuracy, no conflicts would occur. It can be seen that the possible speedup for the instruction port is higher than for the data port. This is due to the higher absolute utilization of the instruction port, which results in more conflicts for false predictions. Also shown are the achieved values of P_{cfa} for $k \in \{0, 8\}$. Again, a system only relying on interleaved memory banks would achieve $P_{\text{cfa}} = 75\,\%$. It can be seen that the blocking mode achieves higher speedups than the FFA mode. In contrast, it was shown that the interval utilization is higher in FFA mode. Hence, there is a trade-off between speedup and interval utilization.

To implement the proposed statistical AIP approach, it is necessary to save the maxima of the pmf functions in a look-up-table (LUT). The observation vector can than be used as an address into the LUT to retrieve the next interval. Shared resources can be pre-allocated accordingly. Fig. 5 shows how many maxima the individual pmf functions yield. Also shown is the number of total memory accesses for the instruction and data port. It can be seen that the number of maxima saturates about four orders of magnitude below the number of accesses. This contradicts the assumption that each transaction is defined by an individual observation vector for $k \to \infty$. In consequence, the implementation overhead is not growing exponentially with k, but only with $\mathcal{O}(\log k)$. Since every maxima can be represented with a single byte (estimations longer than 256 are very rare and can be clipped), the MAP estimator with $k = 15$ can be implemented with around $10\,\text{kB}$ of RAM.

IV. Conclusion and Outlook

In this paper we introduce Access Interval Prediction. We show how statistical information about the program flow can be exploited to reduce conflicts in tightly-coupled memory systems. The performance evaluation shows that over $90\,\%$ of all conflicts can be avoided, using the introduced MAP estimator. In contrast, interleaved memory systems only avoid $75\,\%$ of all conflicts. Further, we highlight a trade-off between achievable speedup and memory utilization. Additionally, the paper estimates the implementation overhead. In future works, we will extended the system to include more CPUs and memory banks.

References

[1] TI, "OMAP 4 mobile applications platform," 2011.
[2] S. Haas *et al.*, "A Heterogeneous SDR MPSoC in 28 nm CMOS for Low-Latency Wireless Applications," in *DAC*. ACM, 2017.
[3] D. Bates *et al.*, "Exploiting tightly-coupled cores," in *SAMOS XIII*. IEEE, 2013.
[4] M. Gautschi, D. Rossi, and L. Benini, "Customizing an open source processor to fit in an ultra-low power cluster with a shared L1 memory," in *GLSVLSI*. ACM, 2014.
[5] A. Rahimi *et al.*, "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters," in *DATE*. IEEE, 2011.
[6] M. Dehyadegari *et al.*, "A tightly-coupled multi-core cluster with shared-memory HW accelerators," in *SAMOS*. IEEE, 2012.

[1] Given a ratio of 1:2 between master port and memory bank.

This document is a preprint of: R. Wittig, M. Hasler, E. Matus et al., "Statistical Access Interval Prediction for Tightly Coupled Memory Systems," in Proceedings of IEEE Symposium on Low-Power and High-Speed Chips (COOLCHIPS 2019), Yokohama, Japan, Apr 2019. DOI:10.1007/978-3-030-27562-4_16

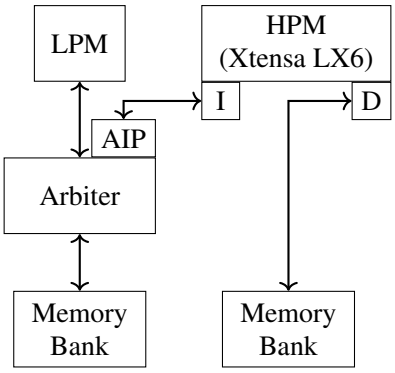


Fig. 1: AIP System Model.

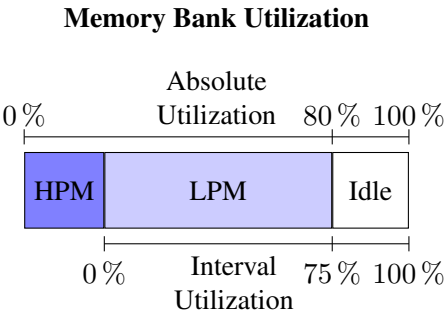


Fig. 2: Difference between *absolute* and *interval* utilization. In this example the absolute utilization sums up to 80 %. The interval utilization is only 75 %.

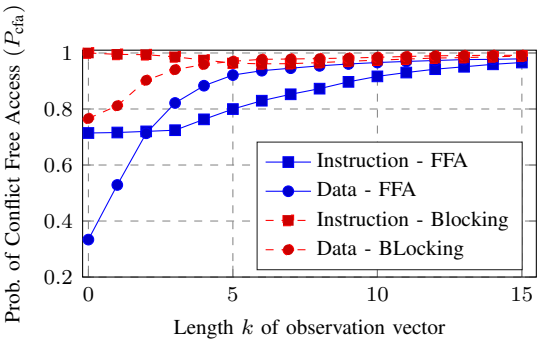


Fig. 3: Probability of conflict free access P_{cfa} for free-for-all (FFA) and blocking mode.

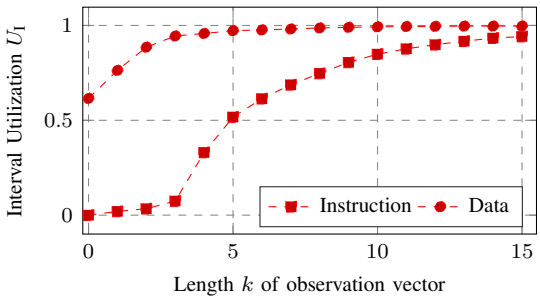


Fig. 4: Interval utilization U_{I} for blocking mode. For free-for-all U_{I} equals 1.

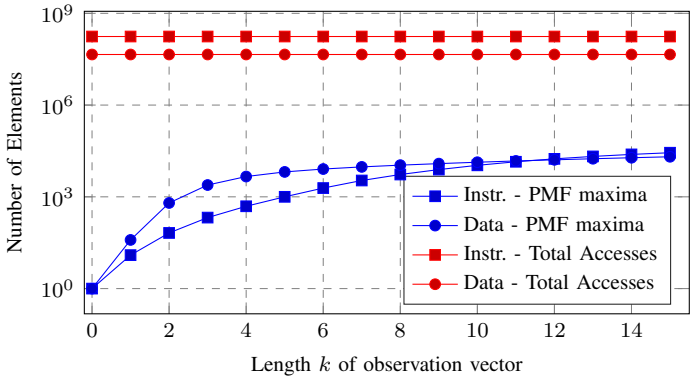
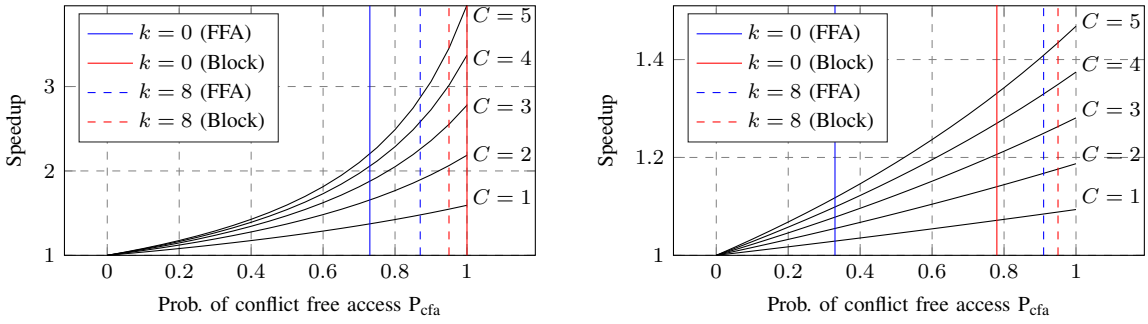


Fig. 5: Number of maxima in probability mass function (PMF) over the size k of the observation vector (average over all testcases).



(a) Instruction port

(b) Data port

Fig. 6: Performance gain of Access Interval Prediction (AIP).