Modeling Accesses to Shared Memories in Multi-Processor Systems-on-Chip

Adam Kostrzewa*, Selma Saidi†, Rolf Ernst*
*Technische Universität Braunschweig, Germany
†Technische Universität Hamburg, Germany

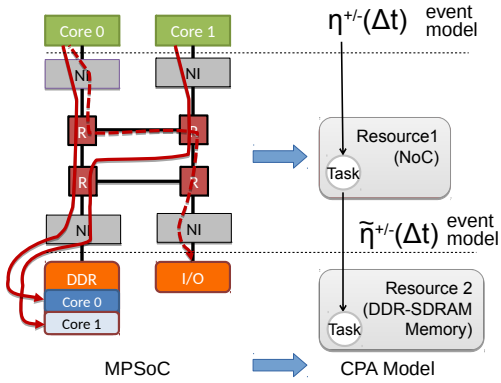

Fig. 1: Model of an MPSoC in CPA with interference resulting from the accesses to shared resources.

I. Introduction

Memory accesses constitute most of the traffic in Multi-Processor Systems-on-Chip (MPSoCs). Therefore, controlling interference on the interconnect and memory becomes a main issue in real-time and safety critical domains. Achieving this goal implies worst-case dimensioning to provide temporal guarantees required by safety standards e.g. ISO26262. However, even with a static task-to-processor mapping, the execution of applications is usually not independent due to the interference on shared resources which couples executions on different processors. In MPSoCs to conduct a single memory access, a task must acquire several resources, e.g. interconnect and memory controller, with independent arbiters and often provided by different vendors. The designer must assure that the effects resulting from coupling these different arbiters will not lead to pessimistic formal guarantees or decreased performance and utilization, which is usually difficult to achieve.

For instance, off-chip DRAMs memories are in particular very sensitive to the locality of accesses. They are organized into different banks to store data and use for each bank an internal caching mechanism which determines whether a new request is accessing an active (i.e open) row leading to small access latencies or if a new row has to be activated leading to larger access latencies. Consequently, the analytical models of MPSoCs must take also the arrival and processing order of memory accesses into account - otherwise only very pessimistic or no guarantees can be given.

II. Models of Accesses

Compositional Performance Analysis (CPA) [1] framework makes use of three main components: resources, tasks, and event models. A model of MPSoCs can be constructed using CPA as depicted in Fig.1, where resources are used for the modeling of processing or network nodes (e.g. CPUs, router ports, control units) and tasks are mapped to resources and compete for the service provided by them. The allocation of the service depends on the selected scheduling policy.

The analysis for an MPSoC system starts with the definition of access models for running applications. To capture the dynamics of the systems behavior, tasks accesses to the memory are abstracted using event models which define arrival functions $\eta^-(\Delta t)$ and $\eta^+(\Delta t)$. These models provide a lower and upper bounds on the number of events (memory accesses) in any half-open time interval. Consequently, they allow to capture all possible event arrival patterns/scenarios within interval bounds. Correspondingly, the minimum and maximum distance functions $\delta^-(n)$ and $\delta^+(n)$ are counterparts of event arrival functions η^+ and η^- respectively defining a lower and upper bound on the time interval between the first and the last event in any sequence of n occurrences of events. An overview of the methods for obtaining typically used behavioral models is available in [1].

Access to memory in MPSoCs are modeled by CPA using a directed graph. In such setup, edges symbolize dependencies and nodes denote tasks consuming service provided by resources (e.g. router's hop latency or processing time of memory controller). Consequently, temporal service of a resource varies per activation between best- and worst-case execution/transmission time. The jitter (the difference between maximum and minimum response times) allows to derive new output event models. An output event model of a task on a particular resource becomes the input event model for its dependent task(s) on another resources e.g. each router in the path influences the input models of its neighbors.

III. Interconnect Analysis and Derived Metrics

In the MPSoCs context, each memory access must firstly traverse the interconnect before it gets served by the memory controller. However, each of them may be further divided into sub-resources (i.e., sub-arbiters). For instance, many modern MPSoCs are equipped with Networks-on-Chip (NoCs). Commonly used wormhole-switched NoCs with multi stage arbitration are not designed to meet the real-time and/or safety requirements but rather to deliver high average case performance. In such networks, ongoing transmissions compete for link bandwidth (output ports) and buffer space (virtual channels). NoCs resources are not reserved in advance, i.e. packets are switched as soon as they arrive, and all traffic receive equal treatment.

Each router is conducting its arbitration locally and independently of each other. Therefore, for obtaining the worst-case end-to-end latencies the CPA is analyzing each router iteratively and propagating the event models (i.e. jitters) along the

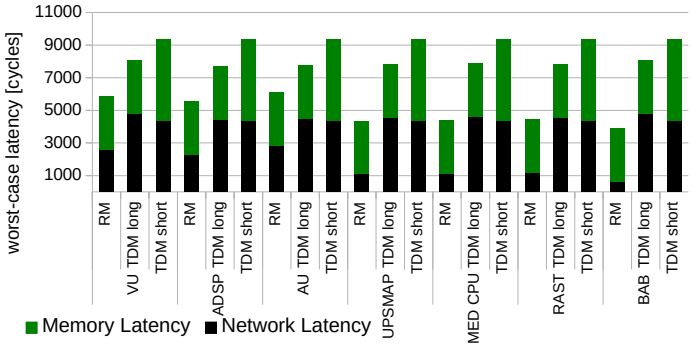

Fig. 2: Effect of memory locality on the total transmission latencies for MPEG-4 module using TDM and RMs, based on [2].

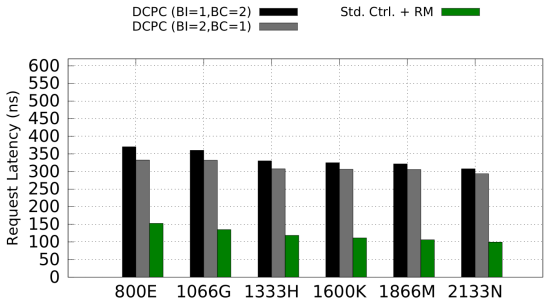

Fig. 3: Worst-case latency for a 256-bytes request on DDR3 devices (a) with 8-bit wide interfaces and (b) with 32-bit wide interfaces, from [3].

path. However, depending on the selected arbitration method traversing a router may require acquiring its several internal resources, see [4]. As an example, in case of the popular iSLIP arbiter, we model each router as a processing system with shared resources. The time necessary for the transfer of a packet through the output port of a router is then modeled as the execution of a task on a processing resource. The arrival of a packet at the input port of the router corresponds to a task activation formally described via event models. Forwarding of a packet requires access to the output port of the router what can be modeled with a mutually exclusive shared resource for every input port of a router. The arbitration scheme used at the output port corresponds to a scheduling policy at the computational resource, while the input arbitration corresponds to a shared resource locking protocol. Consequently, a packet's latency is composed of the time required for

- physical transfer of throughout interference,
- blocking due to other packets stored in buffers (e.g. FIFO),
- blocking due accesses from other input ports to the same output ports,
- backpreassure blocking.

This results in a complex spectrum of direct and indirect interferences between data streams which may endanger the system safety. Therefore we define in [3] a global and dynamic control layer for NoCs which decouples admission control from arbitration in routers. It considers the NoC as a single shared resource and allows to guarantee the access to an entire transmission without any low-level packet interference which simplifies the analysis and also preserves the locality of memory accesses and performs better than commonly used TDM based approach, see Fig2.

IV. Memory Latency

Worst-case memory latency can be obtained following a similar procedure as in the case of the interconnect. Firstly, we must obtain input access arrival curves for memory controller resulting from the *worst-case propagation jitter* of a memory access in an on-chip interconnect denoted by the difference between its worst- and best-case network latency.

The memory scheduling is complex due to the stateful structure of DRAMs. Memory accesses are translated by the memory controller into internal DRAM commands e.g. activate (load a row into a buffer), write (write row to the buffer), read (read a row into the buffer), pre-charge or refresh. Timing depends on the sequence of the commands i.e. history of accesses. Therefore, for achieving the worst-case guarantees two approaches are possible.

Firstly, it is possible to eliminate the correlation between latency and locality of accesses, through the design of memory controller i.e. custom predictable memory controllers. These approaches rely on a close-row policy where a memory row is always precharged (i.e closed) after it has been accessed, thereby releasing the timing dependencies between requests accessing the same bank. The interleaving between memory requests is statically managed where requests are transformed into memory access patterns defined as bundles of reads and writes targeting the same row. However, the close-page policy reduces the performance of the system as it does not take advantage of the DRAM internal level of caching thereby significantly increasing the degree of pessimism for applications where subsequent requests are targeting the same row.

Unlike customized predictable DRAM controllers, commercial-off-the-shelf (COTS) memory controllers in general-purpose systems are optimized for the average-case performance and for this they rely on the open-row policy. Consequently, the analysis must for each memory request consider the locality of accesses. As reported in [2], the introduced control layer allows to provide isolation at the NoC and memory level and to preserve the locality of accesses allowing the use of COTS DRAMs [3]. This combination allows to provide better or comparable performance than close-page policy customized predictable controllers, as depicted in Fig 3.

References

[1] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *IEE Proceedings Computers and Digital Techniques*, 2005.

[2] A. Kostrzewa, S. Saidi, L. Ecco, and R. Ernst, "Dynamic admission control for real-time networks-on-chips," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 719–724, Jan 2016.

[3] A. Kostrzewa, S. Saidi, L. Ecco, and R. Ernst, "Ensuring safety and efficiency in networks-on-chip," *Integration, the VLSI Journal*, vol. 58, no. Supplement C, pp. 571 – 582, 2017.

[4] S. Tobuschat and R. Ernst, "Real-time communication analysis for networks-on-chip with backpressure," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 590–595, March 2017.