Predictable SoC architecture based on COTS multi-core

Nitin Shivaraman
Nanyang Technological University
Email: nshivaraman@ntu.edu.sg

Sriram Vasudevan
Nanyang Technological University
Email: sriram006@e.ntu.edu.sg

Arvind Easwaran
Nanyang Technological University
Email: arvinde@ntu.edu.sg

***Abstract*—With the increasing complexity of real-time embedded applications and the availability of Commercial-Off-The-Shelf (COTS) multi-cores, time-predictable execution on these platforms has become a necessity. However, there are several challenges to achieving this predictability, primarily arising due to hardware resources shared between the cores (memory controllers, caches and shared interconnect).**

In this demo, we present a novel System-on-Chip (SoC) architecture based on COTS multi-cores that address some of these challenges. Specifically, we develop an architecture that enables COTS multi-cores to predictably access external memory. This SoC is designed using hybrid hardware platforms comprising a COTS multi-core and closely coupled Field Programmable Gate Array (FPGA), e.g., Xilinx Zynq ZC706. In our design, the COTS multi-core (ARM Cortex-A9 dual-core) is integrated using a high-speed interconnect with an arbiter module and the Memory Interface Generator (MIG) Xilinx memory controller on the FPGA. Through experiments we show that the proposed architecture has a precisely predictable worst-case memory access latency when compared to a COTS-only design.

I. Introduction

Embedded systems support time-critical and safety-critical functionalities such as those applications in automotive, avionics, medical systems, etc. In such Real-Time Systems (RTS), being able to predict the Worst-Case Execution Time (WCET) of the application on the underlying hardware platform is an essential requirement and we refer to this property as *time-predictability*. In order to support the increasing hardware performance demands of such RTS, multi-core processors seem like a natural choice. One of the primary advantages of using multi-cores is its Size, Weight and Power (SWaP) characteristics making the die fit into a single package. Execution-time of an application running on a core depends on how the application on the other cores uses the shared resources of the multi-core. Additionally, COTS multi-cores are generally preferable in safety-critical applications due to a long service history in a variety of applications. The proposed architecture illustrates the potential of using COTS multi-cores together with closely coupled Programmable Logic (PL) for real-time embedded applications with strict timing requirements. We use Xilinx Zynq ZC706 platform for this demo to illustrate the predictability of the architecture.

II. Related Work

Several studies have been done to obtain predictability in COTS multi-cores. Some of these focused on analytical techniques to estimate the WCET (e.g., [1] and [2]). These techniques have limited applicability due to either unrealistic assumptions or lack of information about the architecture.

There have also been several studies that propose modifications to the hardware such as arbitration schemes for interconnect and controllers (e.g., [3], [4], [5], [6] and [7]). However, they do not address the problem of how COTS multi-cores can be modified to provide such capabilities. Our customizable SoC architecture is built to address this problem. Also, the techniques proposed in the above studies are orthogonal to our work and can be implemented in the PL of our architecture.

Another category of research focuses on building the entire multi-core from scratch on FGPA (e.g., [8]). Considering the SWaP characteristics and service history of COTS multi-cores against FPGA-based designs, our architecture offers the advantage of using as many COTS components as possible with minimal support from FPGAs.

The work by Sha et.al. proposes single core equivalence of multi-core processors by combining different software-level techniques to address predictability challenges of various shared hardware components [9]. Such software-level mechanisms have limitations in terms of the arbitration policies that can be implemented, and may also have substantial implementation overheads.

III. Proposed Architecture

The proposed architecture is shown in Figure 1. In this design, memory requests from the Processing System (PS) are re-routed to the FPGA/PL and eventually sent to the DDR which is connected to the PL. The key idea of this design is that the components that have been identified to cause unpredictability in COTS multi-cores (memory controllers, cache hierarchy and shared interconnect) are disabled and their functionalities are handled separately on the FPGA. This gives us the flexibility to provide custom solutions to handle memory requests in a predictable manner.

In this initial phase of the architecture, we disable caches at all levels as we focus on the feasibility of predictably routing memory requests from PS to PL. As seen from Figure 1, the memory access requests from the cores bypass the Snoop Control Unit (SCU), the L1 and L2 caches and arrives at the master interconnect. As the cores are address mapped to the General Purpose port 1 (GP port physically connects

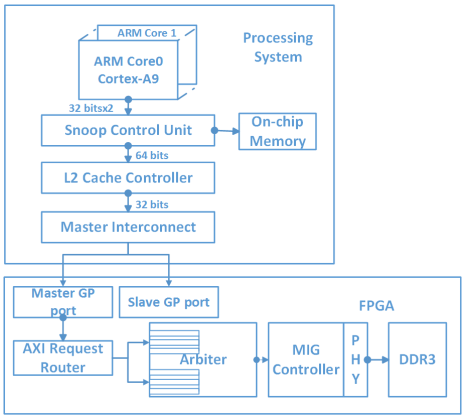

Fig. 1. Predictable SoC architecture

the ARM cores with the FPGA), the request that arrives at the interconnect pass through this port and are received by the First In First Out (FIFO) buffers which are present in the arbiter module of the design. There are two separate buffers which collect the memory requests based on the core that generates the request. The arbiter module implements a round-robin arbitration scheme based on which it selects the memory request from the buffers and sends them to the Memory Interface Generator (MIG) memory controller [10]. This arbiter module manages the scheduling of requests external to the MIG controller thereby providing memory access in a predictable fashion. MIG interfaces to a PHY (physical layer) which in turn connects to the external DDR memory.

IV. Demo

In this demo, we show the predictability of the proposed architecture by performing two different experiments and displaying the output as graphs. The experiment setup would consist of the Xilinx Zynq ZC706 board connected to a host for programming and UART display. The first experiment is to measure the latency for each memory transaction (reads and writes) and the second is to measure the throughput of the architecture. The experiment details are presented below:

1) For single core latency experiments, we perform 100 memory transactions of which every 10 transactions are to a single row. After 10 transactions we switch to a new row. All transactions are performed to the same bank. This would demonstrate the predictable nature of access. We will also be able to observe row switching and refresh in the DDR memory precisely.
2) For all dual core experiments, we shall use core 0 as the observing (measuring) core and core 1 will be the one which causes interference.
3) For dual core latency experiments, we perform 100 memory transactions from core 0 to a single row and a single bank. On the other hand, core 1 will continuously perform memory transactions to the same row and bank in a loop. This is done to make sure that core 0 memory transactions suffer sufficient interference from memory transactions of core 1.
4) For latency experiments, to measure the time for each memory transaction, we initially measure the overhead of the timer (global clock of Zynq 706) by starting and halting the timer without performing any memory transaction. This overhead is subsequently negated from the time measured for all memory transactions.
5) The results of the experiments are plotted as a histogram of latency versus the frequency of occurrence of the latencies. This would show the evidence of the different latencies measured, described in steps 1 and 3.

V. Work in Progress

The main contribution of this work is to demonstrate an architecture which uses COTS multi-core extensively and supported by logic on FPGA to predictably access memory. This initial design has the MIG memory controller which is externally supported by an arbiter module which predictably schedules memory transactions from both the cores. However, the arbiter has an overhead which reduces the throughput of the current architecture. The private and shared caches are also disabled in this architecture.

The next step to this work in progress is to replace the MIG with our custom memory controller solution. The custom memory controller solution will integrate the arbiter unit along with the memory controller which will not only help to improve the throughput but also help to implement fine grain arbitration techniques. The inclusion of a predictable caching architecture is also a part of the future work to make the architecture robust and practical to be used in RTS.

Acknowledgment

This work was funded in part by MoE Tier-1 grant number RG21/13.

References

[1] M. Lv, W. Yi, N. Guan, and G. Yu, "Combining abstract interpretation with model checking for timing analysis of multicore software," in *RTSS, 2010 IEEE 31st*, Nov 2010, pp. 339–349.
[2] D. Hardy and I. Puaut, "WCET analysis of multi-level set-associative instruction caches," *CoRR*, vol. abs/0807.0993, 2008.
[3] M. Paolieri, E. Quiones, F. Cazorla, and M. Valero, "An analyzable memory controller for hard real-time cmps," *Embedded Systems Letters, IEEE*, vol. 1, no. 4, pp. 86–90, Dec 2009.
[4] J. Reineke, I. Liu, H. D. Patel, S. Kim, and E. A. Lee, "Pret dram controller: Bank privatization for predictability and temporal isolation," ser. CODES+ISSS '11, pp. 99–108.
[5] Y. Krishnapillai, Z. P. Wu, and R. Pellizzoni, "A rank-switching, open-row dram controller for time-predictable systems," in *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, July 2014, pp. 27–38.
[6] E. Lakis, "FPGA implementation of a time predictable memory controller for a chip-multiprocessor system," Master's thesis, Technical University of Denmark, DTU, 2013.
[7] B. Akesson, K. Goossens, and M. Ringhofer, "Predator: A predictable sdram memory controller," ser. CODES+ISSS '07, 2007, pp. 251–256.
[8] M. Paolieri, J. Mische, S. Metzlaff, M. Gerdes, E. Quiñones, S. Uhrig, T. Ungerer, and F. J. Cazorla, "A hard real-time capable multi-core smt processor," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 3, Apr. 2013.
[9] L. Sha, M. Caccamo, R. Mancuso, J.-E. kim, M.-K. Yoon, R. Pellizzoni, H. Yun, R. Kegley, D. Perlman, G. Arundale, and R. Bradbord, "Single core equivalent virtual machines for hard real-time computing on multicore processors," Tech. Rep., 2014.
[10] Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions. http://www.xilinx.com/products/intellectual-property/mig.html