Priority division: A high-speed shared-memory bus arbitration with bounded latency

Hardik Shah
ForTISS GmbH
Guerickestr.25, 80805 Munich
Email: shah@fortiss.org

Andreas Raabe
ForTISS GmbH
Guerickestr.25, 80805 Munich
Email: raabe@fortiss.org

Alois Knoll
Institut für Informatik VI
TU München, 85748 Garching
Email: knoll@in.tum.de

***Abstract*—In state-of-the-art multi-processor systems-on-chip (MPSoC), interconnect of processing elements has a major impact on the system's overall average-case and worst-case performance. Moreover, in real-time applications predictability of inter-chip communication latency is imperative for bounding the response time of the overall system. In shared-memory MPSoCs buses are still the prevalent means of on-chip communication for small to medium size chip-multi-processors (CMPs). Still, bus arbitration schemes employed in current architectures either deliver good average-case performance (i.e. maximize bus utilization) or enable tight bounding of worst-case-execution time. This paper presents a shared bus arbitration approach allowing high bus utilization while guaranteeing a fixed bandwidth per time frame to each master. Thus it provides high-performance to both real-time and any-time applications or even a mixture of both.**

The paper includes performance results obtained while executing random traffic on a shared bus implemented on a FPGA. The results show that our approach provides bus utilization close to static priority based arbitration, a fairer bandwidth distribution than Round Robin and latency guarantees identical to TDMA. With this it combines the best properties of these schemes.

I. Introduction

On chip interconnect, such as shared buses [1], [2], [3], heavily influence an application's execution characteristics on a particular architecture. In general, shared buses have a simple architecture which facilitates plug and play use for all the components that comply with the bus protocol. Shared bus architecture contains an arbiter to resolve conflicts when multiple bus masters intend to execute a transfer on the bus simultaneously. Differences in the arbitration schemes lead to differences in the system's timing behavior and influence its suitability for different application domains. Mostly two classes of applications need to be discriminated here. Firstly, best-effort (or any-time) applications (e.g. multimedia applications) for which average-case performance and throughput needs to be maximized. These applications favor arbitration schemes providing highest-possible bus utilization and keep the bus throughput close to the maximum. The second class is (possibly safety-critical) real-time applications. Here, exceeding the time-limit is prohibitive and as such only guaranteed resources can be considered. Arbitration schemes providing deterministic maximum latency lend themselves well to Worst-Case Execution Time (WCET) analysis and allow tight bounds on WCET. In such applications time triggered access to shared resources is mostly applied in multicore architectures [4].

This paper presents an arbitration approach combining benefits of static priority based arbitration with time-division multiple access (TDMA). It allows high bus utilization while guaranteeing a fixed bandwidth per time frame to each master. Experimental results show that the bus utilization is close to static priority based arbitration while guaranteeing bandwidth per time frame to each master on the bus.

II. Related Work

A. Predictable Architectures

A lot of research has been conducted on architectures viable for hard-real time systems. [5] propose a fully deterministic architecture with repeatable timing. Considerable effort is spent on developing pipelines, memory controllers etc. with this property. Thus enabling low WCET and tight bounds on it. Due to the very restrictive claim of repeatability time-triggered arbitration schemes can be used exclusively and thus it is unlikely that the resulting architecture will provide high-performance with respect to average-case timing as well. The propeller architecture is a commercially available multicore processor implementing a round robin (RR) arbitration scheme on the bus connecting shared resources to cores [6]. The XMOS architecture also provides fully deterministic timing requiring static communication resource allocation on a crossbar-switch connecting four processor cores [7]. [8] propose a virtual simple architecture, that runs a real-time application on both, an arbitrarily complex processor core and a simple, WCET analyzable one. If the complex processor exceeds an intermediate deadline of the application the result of the simple core is propagated, otherwise the result of the complex core is. In [9] a superscalar in-order processor is enhanced to provide hard real time capability. Both, [9] and [8] would complement our approach well and could be combined with it into a multicore providing high throughput along with worst-case guarantees. [10], [11] propose reactive microprocessor cores as targets for esterel programs. A different approach is the detailed analysis of standard architectures in order to derive tight bounds on an application's WCET which is running on them. WCET estimation methods became fairly sophisticated and the bounds are permanently improving. An overview can be found in [12]. Nevertheless, state-of-the-art processors constructed for providing good average-case

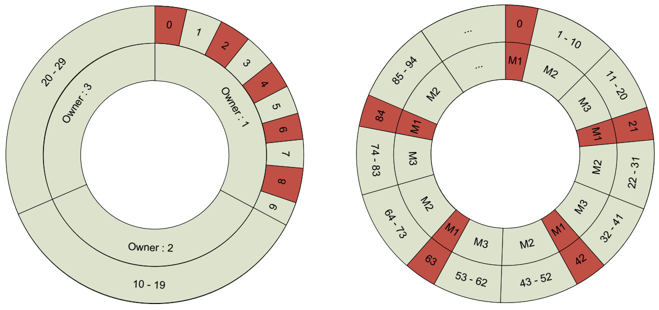
```
x = &location1;                  MOV Rx, &Location1
for (i = 0; i < 5; i++)          LDA R0, Rx
{                                ADI Rx, 100
  y[i] = *x;                     LDA R1, Rx
  x+ = 100;                        ...
}                                ADI Rx,100
                                 LDA R4, Rx
```

Listing 1: Pseudo Code for Non-sequential Access

performance do not lend themselves well towards this type of analysis. Thus, WCET is still largely overestimated.

B. Analysis of Arbitration Schemes

Arbitration schemes are of two classes - static priority based and time division based (e.g. TDMA). Static priority based arbitration provides maximum bus utilization at the cost of excessive latencies to the low priority masters and unpredictability in execution time due to the interference from other bus masters [13]. TDMA based arbitration distributes latency among masters in a predictable manner and provides guaranteed bandwidth to each master at the cost of poor bus utilization. RR arbitration is an improvement of TDMA. It provides high bus utilization by rotating the time slots only among active masters. Thus, when a master does not request bus access or uses only a portion of its time slot, the time slot of next active master is started immediately and the bus is kept busy while there is at least one active master. The RR scheme can lead to excessive latencies in practical cases. E.g., consider a case of an uncached shared-memory architecture with three processor cores. Each CPU is granted a time slot of 10 clock cycles. Considering pipelined transfers, this means each CPU can do maximally 10 transfers during its assigned slot. Listing 1 shows a simple C programme and corresponding assembly code after loop unrolling. It accesses five non-sequential shared memory locations. Fig. 1 depicts the behavior of this code in TDMA and RR schemes. The outer circle in the figure depicts clock cycles and the inner circle depicts the bus owner during those clock cycles. Red colored slots mark executed read transfers. In the TDMA scheme the idle cycles 1,3,5,7 are used to increment the read address as shown in the assembly code. In case of TDMA such execution is completed within 10 clock cycles at an expense of 50% loss in bus utilization. In the RR scheme, CPU2 takes over in the 1^{st} clock cycle because CPU1 is not using the bus. After CPU2 finishes CPU3 is granted bus access in clock cycle 11. In the worst case scenario CPU1 gets the bus ownership back in the 21^{st} clock cycle. At this point CPU1 can do one more transfer and then it has to wait for another 20 clock cycles to get the ownership back. Fig. 1b shows that CPU1 needs 85 clocks to finish its five transfers in this case. The excessive, though predictable, latency increases the upper bound on the WCET. If all CPUs execute bus transfers, the minimal execution time of 15 clock cycles can only be achieved if the slot size equals 1 clock cycle. Making slots only 1 clock cycle big does not allow burst transfers on the bus and thus may degrade performance (e.g. of streaming applications) severely. The relationship of



(a) Execution Time in TDMA (b) Execution Time in RR

Fig. 1: Bus Schedule for code of Listing 1

slot size and burst length will be discussed in more depth in Sec. IV-B.

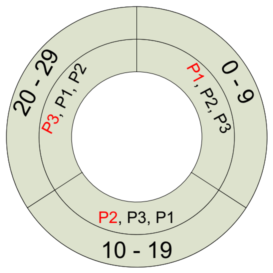
Static priority, TDMA and RR based arbitration schemes employ simple architecture and exhibit high compatibility with existing systems. There have been several attempts to either increase the bus throughput or achieve predictable behavior. In the automotive domain, FlexRay [14] employs a mixture of TDMA and RR by dividing the bus schedule in static and dynamic parts to achieve fairness and high throughput. Using this protocol in MPSoC is not advantageous as it would incur poor bus utilization and unfairness in static and dynamic parts respectively.

In this paper we will focus on performance and predictability of Arbitration Schemes exclusively.

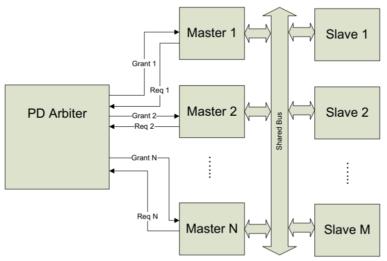
III. Priority Division

Like TDMA, PD divides the bus schedule into a number of slots. Instead of declaring one master as the owner of a particular time slot it assigns each master a priority in the according slot. Thus, each time slot has its own set of priorities for masters. It does not enforce any rule for slot size and priorities. This opens a new dimension of design space exploration where the combined effect of slot sizes and priorities inside slots can be investigated. However, given that no knowledge about the requirements of the applications running on the masters can be obtained the choice of priorities is straight forward. To guarantee bandwidth to all masters, each master must have highest priority in at least one time slot. To evenly distribute bus bandwidth not used by the highest priority master rotating priorities from slot to slot is the simplest solution. Note that, in our scheme the master with highest priority always gets ownership of the bus immediately. Therefore low priority masters are interrupted in this case. Instead of starting the next time slot by rotating the timing wheel like in RR, the highest priority master lends its time slot to a lower priority master only during its idle time. As soon as the highest priority master becomes active, it regains the bus access from the lower priority master. With this choice of priorities the arbitration will behave identical to TDMA under full load and thus give the same lower bound guarantees making it compatible with the existing methods and tools for WCET analysis.

A master can execute burst transfers during the slots in



(a) Priority Assignment in Slots



(b) Basic Architecture

Fig. 2: Priority Division

which it has the highest priority. During other slots burst transfer requests are granted but will be terminated early if a master with higher priority requests bus access.

Fig. 2a depicts a typical slot assignment for a 3 masters system, with an identical slot length of 10 clock cycles for all masters. The inner circle marks priorities in a particular slot in descending order. Fig. 2b depicts the functional architecture. It is identical to traditional bus architectures and thus fully compatible with existing systems.

IV. Test Setup

We tested the proposed arbitration scheme and compared it with static priority, RR and TDMA on a Cyclone III FPGA from Altera. There are three hardware traffic generators capable of generating random traffic and an on-chip shared memory connected to the shared bus.

A. Bus Utilization Test

In this test, each master generates a random traffic to the shared memory. Traffic patterns are of three categories: light, high and aggressive. Each master executes T random transfers with identical T_{avg}. Between two consecutive transfers each master waits for ω random number of clock cycles. The ratio of the average number of wait clock cycles, ω_{avg}, to T_{avg} determines the traffic category and is given by the following equations for a three masters system,

$$\text{light traffic: } \omega_{avg}/T_{avg} = 2 \tag{1}$$
$$\text{heavy traffic: } \omega_{avg}/T_{avg} = 1.5 \tag{2}$$
$$\text{aggressive traffic: } \omega_{avg}/T_{avg} = 1 \tag{3}$$

For TDMA and PD, each master is assigned a fixed slot size of T_{avg} clock cycles. For static priority based arbitration priority of master 1 is highest and of master 3 is lowest. For RR arbitration, slot size of 1 as well as slot size of T_{avg} have been investigated. The results of the test are shown in Fig. 3. We ran the execution for 200 μs and took the measurements of latency and bus utilization. Top graphs in Fig. 3 show the latency of each master and bottom graphs depict the bus utilization. Latency is defined as the number of clock cycles a master has to wait to get the bus ownership. Note that, although static priority, RR and PD all keep the bus busy if at least one master is ready to transfer, the bus utilization is lower for RR and PD compared to static priority. In static priority based

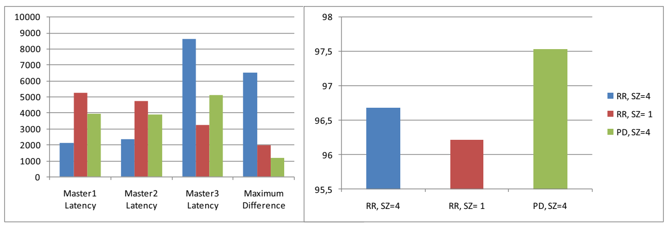


Fig. 4: (a) Latency in Clock Cycles (b) Bus Utilization in %

arbitration bus utilization is high as the arbiter keeps stalling the lower priority masters until the higher priority master goes idle and obliging lower priority masters to overlap idle cycles of higher priority masters by their active cycles. This comes at the cost of excessive latency to lower priority masters. Being a hybrid of static priority based scheme and TDMA, PD partially obliges lower priority masters to overlap idle cycles of high priority masters due to the prioritization within a slot. But to provide bandwidth guarantees to all masters, it also allows active cycles of one master to overlap active cycles of other masters if a new time slot starts. Thus, bus utilization is lower in PD compared to the static priority based arbitration scheme. Considering bus utilization, RR has nearly the same behaviour as PD. Idle cycles of a master are overlapped by active cycles of the next master in the ring as well as active cycle of a master are overlapped by active cycles of the next master if a new time slot starts. Thus, in Fig. 3 bus utilization is similar for priority division and RR schemes. In TDMA, idle cycles of one master are overlapped by an active master only if the time slot of the idle master ends and active master begins. This leads to poor bus utilization as can be seen in Fig. 3.

B. Fairness Test

As discussed in Sec. II-B latency in acquiring bus access of the RR scheme is fair to some masters and unfair to others depending upon their access patterns. A fair arbitration scheme should distribute the available bus bandwidth equally among masters under varying traffic patterns. In this test we used aggressive traffic pattern of equation (3) for all masters. Here, we used $T^1_{fix} = 4$, $T^2_{fix} = 4$, $T^3_{fix} = 1$, $\omega^1_{avg} = 4$, $\omega^2_{avg} = 4$ and $\omega^3_{fix} = 1$ to simulate the example shown in Fig. 1, where T^i and ω^i are active and idle clock cycles of i^{th} traffic generator respectively. Here, all masters generate equal average traffic, only the access pattern differs. Master 1 and 2 generate 4 beat bursts and wait for a random number of clock cycles such that $\omega_{avg} = 4$. Master 3 generates uniform traffic. Since all masters generate equal amount of average traffic, the maximum difference between latencies experienced by masters is the figure of merit. The lower the maximum difference, the fairer the arbitration scheme. In Fig. 4 results of a 0.5 ms execution are shown. The graphs in the figure show latency and bus utilization using the RR scheme, with slot sizes 1 and 4 clock cycles, and PD with a slot size of 4 clock cycle. We can conclude from the graphs that the RR scheme becomes unfair when employing slots as big as 4 clock cycles. Excessive latency experienced by master 3 increases the WCET of the application that generates uniform traffic

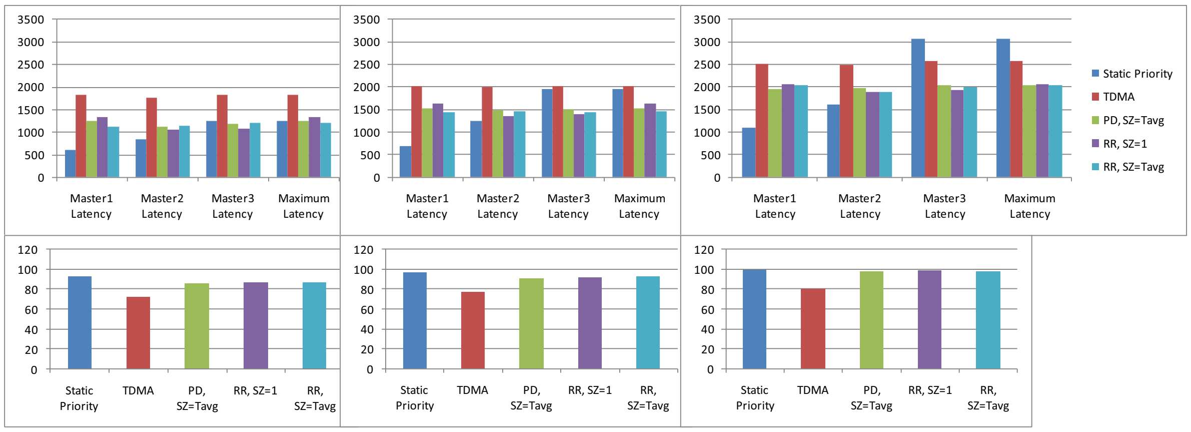


Fig. 3: Latency in Clock Cycles(top) and Bus Utilization(bottom) in %. (a) Low traffic. (b) High Traffic. (c) Aggressive traffic

as discussed in Sec. II. Slot size 1 provides better bandwidth allocation to uniform traffic but bandwidth allocation to burst traffic is degraded.

PD distributes the bandwidth more fairly and is less sensitive to the traffic pattern variations while providing equally high bus utilization.

C. Synthesis

Table I shows resources utilized by our FPGA implementations of the different arbitration schemes for Altera Cyclone III FPGA at 50 MHz frequency. Resource utilization includes arbitration policy and bus multiplexers. Although PD combines the benefits of static priority based arbitration with TDMA it only leads to a minor increase in chip area.

	Total LEs	Registers
Static Priority	119	3
TDMA	130	7
RR	141	7
PD	135	10

TABLE I: Synthesis Results

V. Conclusion and Future Work

In this paper we proposed a novel arbitration scheme for shared buses called priority division. It guarantees bandwidth per time frame to all connected masters while providing high bus utilization. Results obtained by generating random traffic on shared bus were presented showing that our approach outperforms TDMA and provides latencies similar to Round Robin arbitrated buses for random traffic. They also show that it provides much better bandwidth distribution to access patterns mixing uniform traffic and burst traffic than the Round Robin scheme. If some masters make little use of the bus it also provides a more fair distribution of bandwidth to the remaining masters. This renders it suitable for high density multimedia traffic as well as latency sensitive safety critical traffic. Synthesis results for the Altera Cyclone III FPGA were presented that show only a minor hardware overhead is incurred by our approach. Our next step will be the inclusion of support for slave response such as SPLIT within priority division. A study on the performance and power consumption of priority division in real world applications is already under investigation. Furthermore we will investigate on memory hierarchies that go together well with this type of arbitration.

References

[1] "Advance microcontroller bus architecture (amba)." [Online]. Available: http://www.arm.com/

[2] "Coreconnect bus architecture." [Online]. Available: http://www.IBM.com/

[3] "Stbus communication system concept and definitions." [Online]. Available: http://www.st.com/stonline/books/pdf/docs/14178.pdf

[4] M. Schoeberl, "Time-predictable computer architecture," *EURASIP J. Embedded Syst.*, vol. 2009, pp. 1–17, 2009.

[5] B. Lickly, I. Liu, S. Kim, H. D. Patel, S. A. Edwards, and E. A. Lee, "Predictable programming on a precision timed architecture," in *CASES '08: Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*. New York, NY, USA: ACM, 2008, pp. 137–146.

[6] [Online]. Available: http://www.parallax.com/propeller/

[7] [Online]. Available: http://www.xmos.com/

[8] A. Anantaraman, K. Seth, K. Patil, E. Rotenberg, and F. Mueller, "Virtual simple architecture (visa): Exceeding the complexity limit in safe real-time systems," in *In International Symposium on Computer Architecture*, 2003, pp. 250–261.

[9] J. Mische, I. Guliashvili, S. Uhrig, and T. Ungerer, "How to enhance a superscalar processor to provide hard real-time capable in-order smt," *in 23rd International Conference on Architecture of Computing Systems*, pp. 2–14, 2010.

[10] X. Li, J. Lukoschus, M. Boldt, M. Harder, and R. von Hanxleden, "An esterel processor with full preemption support and its worst case reaction time analysis," in *CASES '05: Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems*. New York, NY, USA: ACM, 2005, pp. 225–236.

[11] Z. Salcic, D. Hui, P. Roop, and M. Biglari-Abhari, "Remic: design of a reactive embedded microprocessor core," in *ASP-DAC '05: Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. New York, NY, USA: ACM, 2005, pp. 977–981.

[12] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 1–53, 2008.

[13] R. Pellizzoni and M. Caccamo, "Impact of peripheral-processor interference on wcet analysis of real-time embedded systems," *Computers, IEEE Transactions on*, vol. 59, no. 3, pp. 400 –415, march 2010.

[14] "Flexray communications system." [Online]. Available: http://www.flexray.com/