Mapping And Verification Effort Of Real-Time Application System-On-Chip Memory Controller.

Sindhubala.M [1], Allan Mary George [2],Sneha Susan Abraham [3],
Assistant professor[1,2,3],
Saveetha School of engineering.

ABSTRACT

The increase in number of applications will make the contemporary System-On-Chip more and more complex. These tasks will need multiple processors, share memory hierarchies to perform a particular work before the deadline; this will guarantee its functional correctness. Satisfying all the real-time requirements is very tough. This paper addresses the limitations by proposing a predictable and composable memory controller.
This memory controller will be individualized into front-end and back-end. The back-end memory uses pre-computed memory patterns. The front-end will be predictable arbiters.

Keywords – composable memory controller, memory controller, patterns.

1. INTRODUCTION

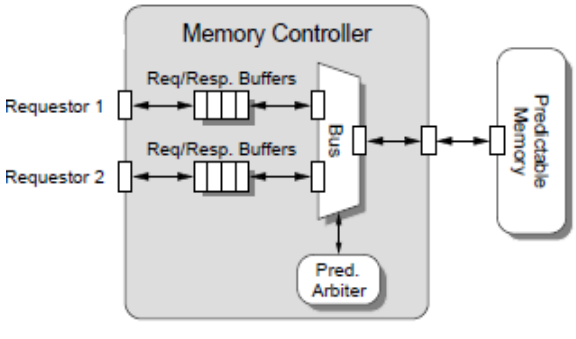
When the systems become more integrate and in more functionality, they will be more and more complex to produce. The manufacturing companies have to prepare this in a short time to stay in the competition. The most difficult problem is to produce the timing requirements. The application software which is in the embedded systems is increasing fastly. Other than the functionality and time requirements, it also needs latency requirements. Previously, there was multi chip circuit board, now even when the number of chip increases it has been integrated into a single chip. This reduces not only the transistor number but also power and cost. According to moore's law, the number of transistors doubles every 24 months. This results in an exponentially increase in hardware productivity gap. This was a major issue in design challenges.

A real-time requirement is divided into hard, firm and soft real-time requirements. Contemporary platforms provide good balance between performance, power consumption, cost and flexibility that is, hard, firm and soft real-time requirements. The problem in this thesis is to design one memory controller which will satisfy all these three requirements, so that it will reduce mapping and verification effort. We need four requirements to achieve this goal; they are predictable, abstraction, composable and automation.

1.1. PROPOSED SOLUTIONS

1.1.1. PREDICTABILITY

This section explains how the



memory controller provides useful bounds and latency of memory transactions. This is Figure 1. overview of predictable memory controller.

approached by combining the memories and arbiters with the behaviors which are predictable. If it have a unshared memory the time will not be managed, so we require a memory called predictable memory, along Figure1. with that we need a predictable arbiter to interfere with the schedule. This will have delay timings, so our approach is to design a general memory controller combining these two. Figure1. gives us an overview of predictable memory controller.

1.1.1.1. PREDICTABLE SDRAM BACK-END.

The requirements of memory controllers will be satisfied by stateless SRAM memories. The behavior is determined by the SDRAM commands. The memory pattern will be in five different patterns , read , write, read/write switching, write/read switching, refresh. Our approach is implemented as SDRAM back-end, as shown in figure 2.

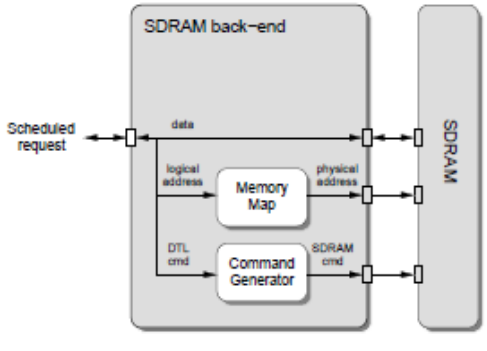


Figure 2. Overview of the predictable SDRAM back-end.

1.1.1.2. PREDICTABLE ARBITRATION

The predictable memory will need predictable arbiter. TDM is our enhanced arbiter. The rate regulator and static priority scheduler combination makes arbiter predictable.

1.1.2. ABSTRACTION

A Latency-Rate server is required to use a common abstraction. This will capture the temporal behavior of different kinds of memory and arbiter types. Theoretically all the arbiters which are predictable belong to Latency-Rate server. This LR model uses two different parameters, service latency and allocated bandwidth. One benefit of this server is that it supports data-flow analysis approach.

1.1.3. COMPOSABILITY.

Some of the applications are too complex to model, so it has to be verified. In order to reduce verification, our controller provides composable services. For composing, that application has to be independent in both value and time domine. There are three composable system design approaches, not sharing any resources, statically schedule all interactions, to share the obtained resources dynamically using TDM during run-time.

Here, we also introduce a fourth approach based on LR server abstraction. This approach extends beyond resources that are inherently composable. It does not have any restriction on applications. It can be dynamically enabled or can be disabled as per the requestor during the run-time by turning on or off the emulation of worst-case interference.

This is implemented by adding a delay block along with the request and response buffers, this contains the additional functionality to implement the composability.

This figure 3 shows an instance of a predictable and composable SDRAM memory controller with two requestors.

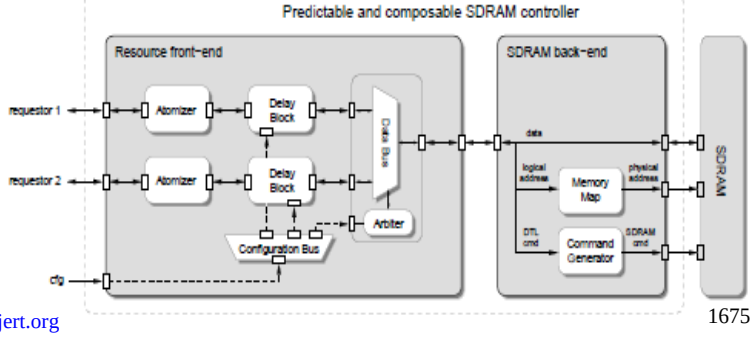


Figure 3. A predictable SDRAM controller supporting two requestors.

To provide this composability service, this delay block will need the information about the maximum interference that can be requested. This will be different for all so a configuration bus is added to the architecture which allows the worst-case interference to get programmed.

1.1.4. AUTOMATION

The memory controller will need automated approach to find the intellectual property parameters. To satisfy this need we designed a automated configuration flow which is shown in figure 4.

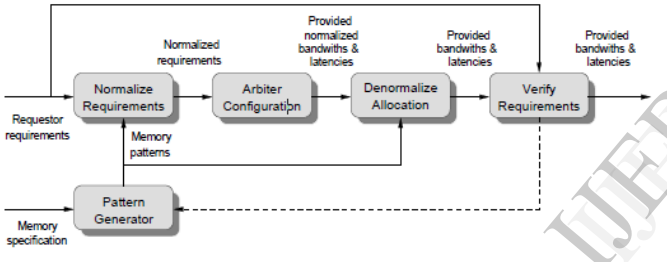


Figure 4. Simplified overview of the automated configuration flow.

This is the design process done by tools. This is an essential part because it reduces the design time, which is a direct impacting time to market.

CONCLUSION

There will be a growth in mapping and verification problem in SoCs, because there is a tremendous increase in number of applications with real-time requirements. So this problem is addressed by designing a memory controller which has the requirements on predictability, abstraction, composability and automation. This is a simple technique which fits in the current type of architecture to let in the back-end benefit from the locality which will reduce power. This also leads to improved arbiter configuration.

REFERENCES

[1] J Reineke, I Liu, Hd Patel, S Kim, Ea Lee," Pret Dram Controller: Bank Privatization For Predictability And Temporal Isolation", Proceedings Of The Seventh Ieee/Acm/Ifip International Conference On Hardware/Software Code Sign And System Synthesis Pages 99-108, 2011

[2] A Hansson, M Ekerhult, A Molnos, A Milutinovic, "Design and Implementation of An Operating System For Composable Processor Sharing", Special Issue On Network-On-Chip Architectures And Design Methodologies 2011.

[3] B Akesson, A Hansson, "Composable Resource Sharing Based On Latency-Rate Servers", Digital System Design, Architectures, Methods and Tools, Dsd '09. 12th Euro Micro Conference, 2009

[4] A Agarwal, C Iskander, "Survey Of Network On Chip (Noc) Architectures & Contributions", Journal Of Engineering, 2009, Scientificjournals.Org

[5] A Nelson, A Molnos, K Goossens, "Composable Power Management with Energy And Power Budgets Per Application", Computer Systems, 2011 - Ieeexplore.Ieee.Org

[6] E Carara, Gm Almeida, G Sassatelli, "Achieving Composability In Noc-Based Mpsocs Through Qos Management At Software Level", Design, Automation & Test In Europe Conference & Exhibition (Date), 2011- Ieeexplore.Ieee.Org