2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Early Performance Exploration of Memory based on JEDEC Specifications

Parvinder Kaur
R&D Engineer, Staff
asija@synopsys.com
Synopsys, Noida 7597740469

Aman Gupta
R&D Engineer, Staff
gaman@synopsys.com
Synopsys, Noida 8058947988

***Abstract*- There are various challenges faced by chip architects ranging from deciding on a suitable memory type for the system to configuring the selected memory to meet the performance requirements of the system. The magnitude of these problems increases with new memory standards getting Introduced and older ones getting updated frequently. Availability of RTL for these updated standards often takes time which leads to further delays in finalizing over the chip designs. This paper presents a SystemC based Generic memory controller TLM model which can be updated much faster using the available specifications and gives the estimates about the impact of updating the memory in the system months before RTL is available. Model also supports a wide range of memory standards which can help the designers in deciding upon the right memory subsystem to use in their chips based on data driven analysis. This paper also presents a case study of how the model can be utilized for adding LPDDR6 memory based on specifications and used for performance exploration in a simple SoC design.**

Keywords-TLM, SystemC modelling, Memory Controllers, Performance exploration, Architecture exploration

.

I. INTRODUCTION

The memory industry is constantly evolving, new memory standards are introduced as frequently as older ones get updated. However, the turnaround time for getting the hardware for these newer versions is extremely high. The need-to-know performance implications of updating memory in the SoC cannot wait for the RTL to be available. Even after RTL is available it typically takes at least another release cycle to update the configurations of other connected components to best utilize the memory. This paper presents a novel solution using SystemC based TLM models which can be updated to visualize the performance impact of new memory months before the availability of RTL. The model requires the timing and other information about any major architectural additions which can be extracted using JEDEC Specifications which are available before RTL. Based on these timings, the model can run simulations for a variety of traffic patterns and provide various visual performance analysis. These performance analysis results can used by SoC developers, who can make early, proactive intelligent decisions on the implications that updating the memory would have, and necessary actions can be taken much earlier than an actual upgrade happens.

When designing an SoC containing DDR memory it is always a challenge to decide the right model of DDR required. In addition, DDR itself offers many configuration options that it becomes challenging to find an optimized memory configuration. Our solution also addresses this challenge by utilizing a highly configurable SystemC model which can be finetuned to mimic most of the commonly used DDR's. The designer can explore for the optimized configuration of the controller using the available simulation capabilities.

This model also caters to the requirements of memory developers due to its independent modular design wherein each module is highly configurable, or even completely replaceable with custom implementations. This makes the TLM model more aligned with the memory designers RTL, with the additional benefits of architecture exploration based on data driven analysis results.

In this paper we will look at a case study of performance exploration of a simple SoC design with an LPDDR6 memory which is yet to be released. Since the JEDEC specification was not available at the time of this project we have used internal ballots which proposes the architecture specifications for LPDDR6. We will also briefly cover the customizations available in the model to mimic the actual RTL.

II. Related Work

The semiconductor industry has successfully adopted SystemC-based Transaction Level Modelling for pre-RTL performance analysis of complex SoC architectures to optimize memory architecture. Optimizing performance has always been an important research area, with studies focusing on power, timing, and other related factors [1][2][3]. While many SystemC-based models exist in the market that can be used to visualize the efficiency of memory, they only provide performance numbers such as utilization, throughput and latency with little information on other areas of improvement. This solution focuses on showing the difference in performance between designs using different DRAM's and evaluating performance with newer versions of memory.

III. Solution Overview

To explore different memories available, we created a SystemC based generic controller model which can be easily extended to include newer memories. The model is highly configurable which enables its use for architecture exploration cases. The model can simulate various static and dynamic latencies like activate to read/write, read to read, read to write, etc. as well as the scheduling of commands based on various scheduling algorithms. Based on these simulations the output provides detailed graphs and charts about key performance metrices like throughput, utilization etc.

The model is integrated into Synopsys tool Platform Architect [4], which provides an integrated environment to create the platform, generate required workload and then run the simulations. Tool also provides a variety of analysis views which gives detailed information about the performance of SoC. This tool offers a variety of integrated systemC models which include IP specific as well as generic models for various components of system like CPU, Interconnect, Memory, Caches etc. which can be utilized to create a virtual prototype of the SoC which is very close to RTL.

Currently the model supports the DDR memory series up to DDR5, LPDDR series up to LPDDR5, GDDR6, HBM2, and HBM3. One can model data-accurate transactions for all sorts of traffic using this SystemC based model. This 'Generic' Memory Controller, which is highly configurable and customizable, can be set up to imitate the features found in various memory controllers. It is not modelled against any specific RTL IP and thus can be customized to support most of the commonly used controllers.

The model offers configurable parameters for all the supported features, queue properties etc. which can be utilized to find an optimized configuration for the controller. These parameters can be swept upon for various traffic to find an optimal value considering all traffic of interest. The address mapping to be used is also fully configurable. This level of configurability helps in finding an optimized controller configuration for any supported memory type [5]. The following figure demonstrates a few of these configurable parameters and address mapping registers.

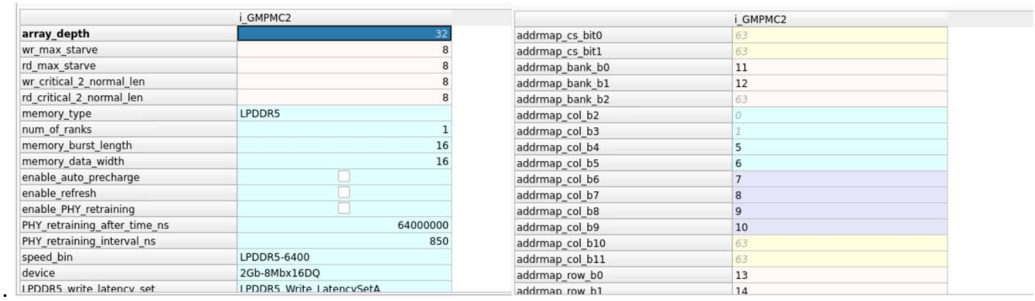

Fig 1 : Parameter view for controller

Generic Controller Architecture

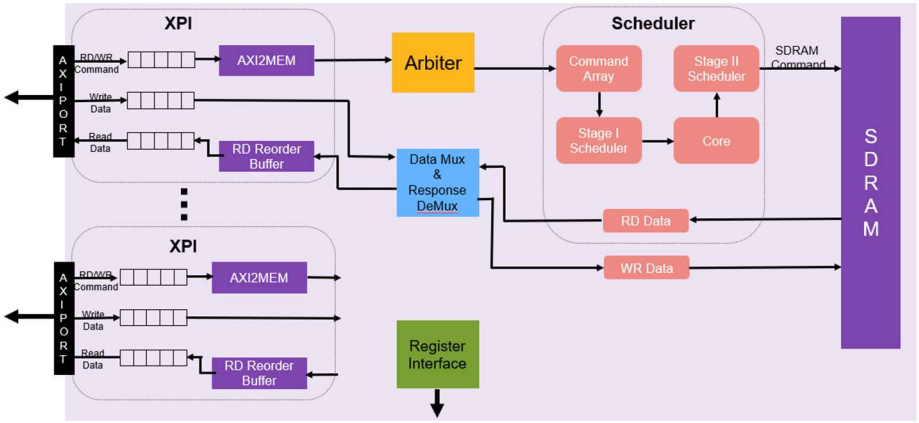

Fig. 2: Generic Memory Controller Architecture

The Port Interface is the starting point of the model. This module accepts AXI as well as CHI transactions. This module also implements configurable queues for handling of various channels of input and output.

The model supports up to 32 ports to accept the input. To arbitrate between multiple ports a round-robin based arbiter is used to forward the requests. These requests are then forwarded to a 2-stage scheduler which grants the transactions one by one based on fixed command priority.

There are separate read and write buffers which support frequency and data width conversion to make the model more configurable. Register Interface further enhances the configurability of the model by allowing the programming of all the registers used inside the model.

Customization Options

The model provides options to replace the major modules like scheduler and arbiter with one's own custom implemented SystemC block.

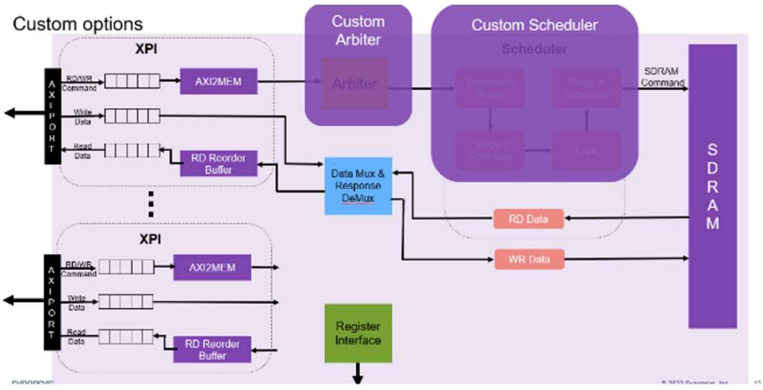

Fig 3: Custom block options

Custom Scheduler: There are various parameters on the model related to CAM sizes, scheduling pattern, etc. which can be configured based upon the actual hardware configurations to be used in the SoC. There is also an option to completely replace the scheduler with custom implementation. This custom scheduler can implement an entirely different scheduling method, for instance in LPDDR6 there can be a custom block implemented to schedule commands in multiple phases of a single clock cycle because of higher speed-bin's support. Even different scheduler channels can be configured with their own unique scheduling algorithms. The Custom scheduler can be implemented as a standalone SytemC module which can be connected to the controller via an interface. The controller would interact with the custom block via these interface ports as shown below.

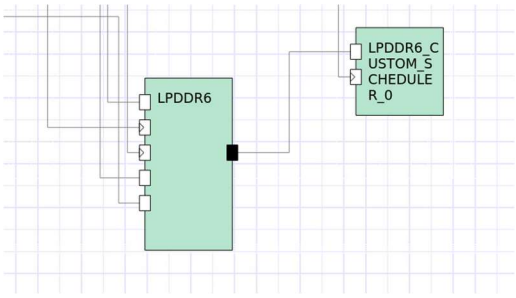

Fig 4: Connections for custom scheduler blocks

Custom Arbiter: The order of access requests can be decided by the custom arbiter using a variety of arbitration algorithms, including round-robin, and priority-based. When making arbitration judgments, it is important to consider elements like priority levels (QOS) - HPR, LPR, VPR, data dependencies, etc. There are request and winner ports available for custom arbiter as shown below. The requests received at AXI ports are forwarded to the respective requestor port and input from the custom block is fed to the winner port which is then forwarded to the scheduler.

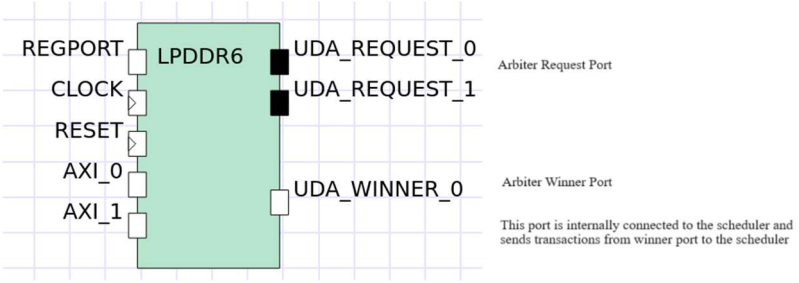

Fig 5: Connections for custom arbiter block

One should analyze the system architecture, workload characteristics, and performance objectives before building and implementing bespoke schedulers and arbiters.

To get the best results for design exploration, the model comes with default values of timings programmed for supported speed-bins and devices. These timings are based on the JEDEC specification for memory, if any customization is required based on RTL, it can be easily done using the register programming interface that the model supports. For example, the following table denotes the common timing constraints for page hit and page miss scenarios which can be provided in the model.

All the constraints used below like T_RL, T_WL are programmed to default as specified in the JEDEC specifications of memories. These can also be programmed to custom values using register port interface which makes the model more customizable to support any device and speed-bin requirements.

Incoming Memory Command	Page Status	Latency Between Memory RD/WR Command and Availability of First Data.
Read	HIT	T_RL
Write	HIT	T_WL
Read	EMPTY	T_RCD + T_RL
Write	EMPTY	T_RCD + T_WL
Read	MISS	T_RP + T_RCD + T_RL
Write	MISS	T_RP + T_RCD + T_WL

Fig 6: Command latencies overview

Performance Analysis

The model also comes with various analysis views which gives a deep insight into transactions path, scheduling of commands etc. The most common and useful analysis views are:

Path Analysis: This view provides statistics for KPI numbers like throughput, latency, utilization, etc. on different time intervals. For instance, the following screenshot denotes the average read latency over a period in intervals of 1 ms.

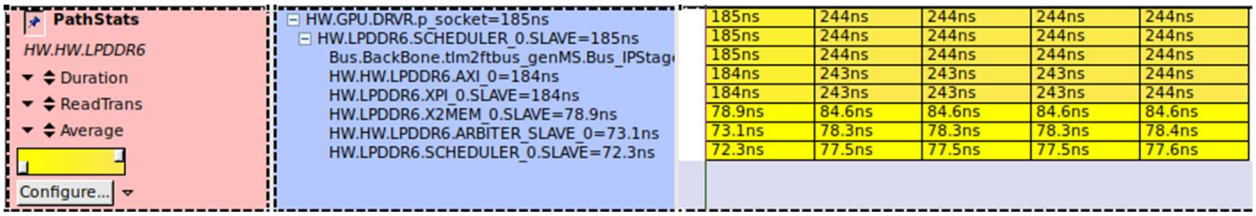

Fig 7: Latency view

There are other detailed traces as well which give the internal order of command scheduling, data flow, buffer utilizations, etc. which can be utilized to optimize the configuration.

For assisting in finding an optimized address mapping, the model provides a hot bit analysis view which gives the details of which bit is toggled how many times. This information can help in finding the optimal address mapping configuration for the controller. The following screenshot shows an example of hot bit analysis view.

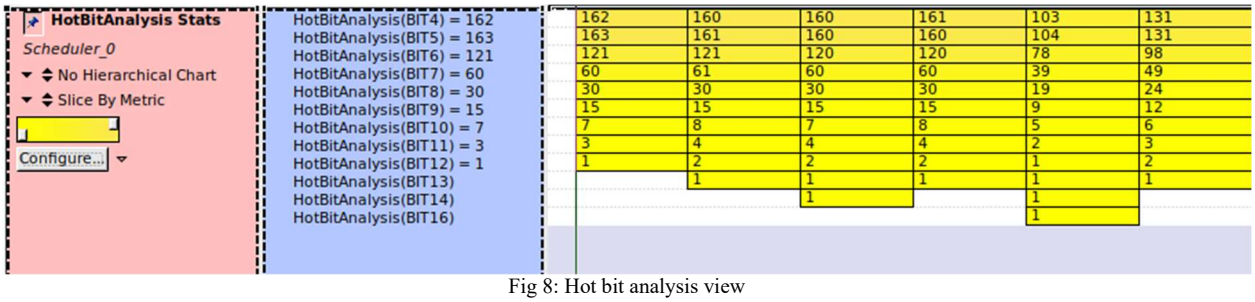

Fig 8: Hot bit analysis view

IV. APPLICATION

In the above-mentioned Generic Controller, we added support for LPDDR6 memory based on JEDEC ballot specifications. The major architecture change was adding support for a burst length of 24. This includes extending the data phase for any transaction to 3 cycles. We also added timings required for LPDDR6. This was done in accordance with the referenced ballot specification.

For this study we created an example SoC with multiple traffic generators (CPUs) which act as initiators and on the target side we have our Generic model configured with the LPDDR6 memory type. We will be using the same configurations with the existing LPDDR5 controller to evaluate the performance differences between both memories. The two CPUs are mapped to a workload [4][6] which can generate the following traffic.

Sequential traffic: This workload generates read transactions followed by write transactions with linear addresses starting from a base provided as parameter input.

Random traffic: This workload generates pure random traffic such that every next address is not linear with the previous one.

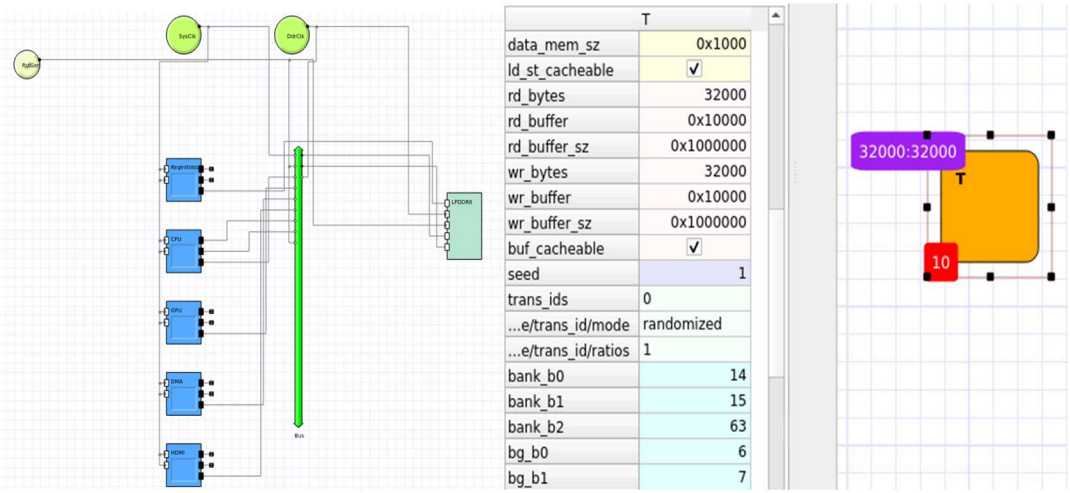

Fig 9: Platform diagram for test platform

V. RESULTS

In this study, we examined throughput differences observed in above mentioned traffic scenarios for a single channel 2 port 16-bit DDR controller. The LPDDR5 controller is programmed at burst length 16 and the LPDDR6 controller is programmed at burst length 24 to comply with the architecture defined in the ballot used.

To observe the difference in burst length, there are data channel utilization traces available in which each data phase in the LPDDR6 platform takes 3 cycles as compared to 2 cycles in LPDDR5. There are command traces available to provide details of commands scheduled on the time axis which reflects the timing changes between the two memories. From following command channel traces, it is observed that write to write timing difference is 5 cycles in LPDDR5 compared to 3 cycles in LPDDR6, which is particular to the speed-bins and devices used in these platforms and in accordance with referenced ballot specification.

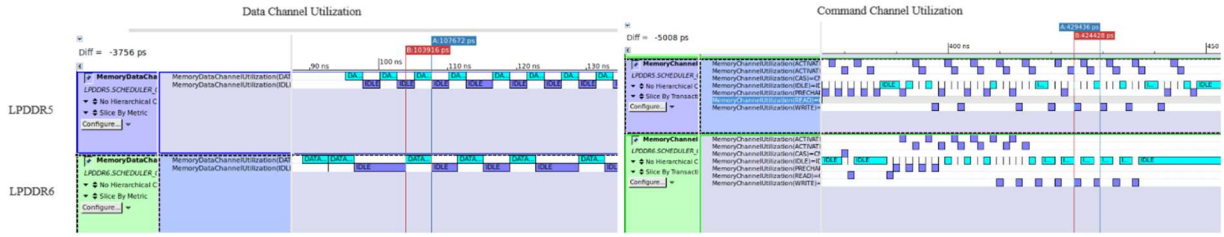

Fig 10: Data channel and command channel traces comparison

The following charts demonstrate the observed throughput in both the above-mentioned scenarios for write and read transactions. These blocks show the observation for platform where the Generic controller is configured with LPDDR5 and LPDDR6.

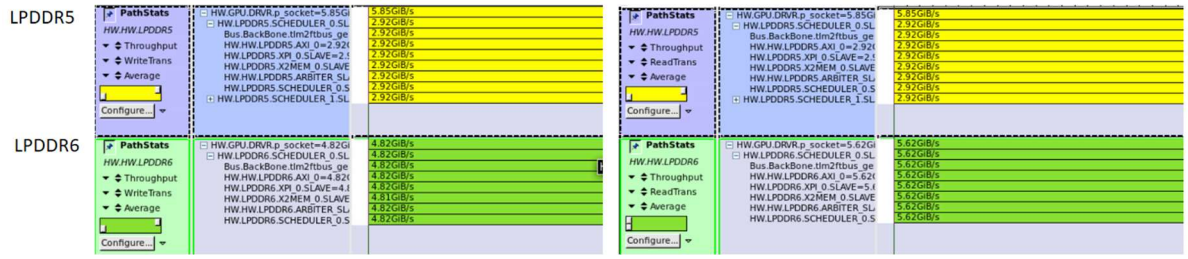

Fig 11: Throughput comparison in sequential traffic

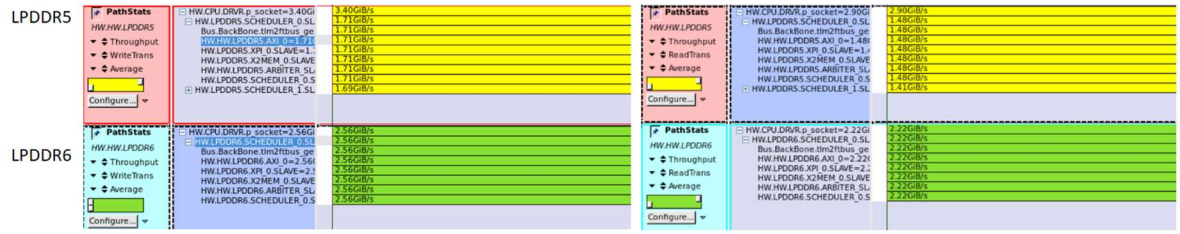

Fig 12: Throughput comparison in random traffic

There are similar and more detailed charts available for analyzing all the major KPI metrices like latency, utilization etc.

The above results conclude that with the current configurations, the LPDDR6 controller, has around a 67% increase in throughput for sequential traffic and around a 54% increase for random traffic w.r.t LPDDR5 controller. In such scenarios, users can leverage detailed analysis like transaction traces, address bits heatmaps, buffer utilization, etc. and modify the controller configuration with optimum values to meet the required KPI metrices. Users can explore for optimum values by sweeping over parameters and creating multiple scenarios with same workload. Tool provides with graphical comparisons for performance numbers which help in identifying the optimum values which meets all KPI requirements of the system.

VI. CONCLUSIONS

Based on the studies conducted it can be concluded that a Generic Controller can be easily leveraged to support newer memory standards based on JEDEC specifications, ballot specifications, etc. which can accelerate performance exploration studies.

Besides exploring for the performance of newer memories, the Generic Controller can also be used for other applications like finding the optimized configuration for memory in a SoC, design exploration based on data analysis, etc.

VII. REFERENCES

[1] IEEE Standard for Standard SystemC® Language Reference Manual by Design Automation Standards Committee of the IEEE Computer Society

[2] Ghenassia, Frank. "Transaction-level modeling with SystemC". Springer, 2005.

[3] T. Grotker, S. Liao, G. Martin, S. Swan. ¨ System Design with SystemC. Kluwer Academic Publishers, 2002

[4] Synopsys Platform Architect for SoC architecture analysis and optimization for performance and power (available at: https://www.synopsys.com/verification/virtual-prototyping/platformarchitect.html)

[5] G. Srivastava, P. Kaur, P. Kaushik and M. Gupta, "Optimizing High Bandwidth Memory in Multi-Die Systems," *2023 IEEE Women in Technology Conference (WINTECHCON)*, Bangalore, India, 2023, pp. 1-5, doi: 10.1109/WINTECHCON58518.2023.10276441.

[6] T. Kempf, M. Doerper, R. Leupers, G. Ascheid, H. Meyr, T. Kogel, B.Vanthournout, "A modular simulation framework for spatial and temporal task mapping onto multi-processor SoC platforms", Design, Automation and Test in Europe.