

Traffic Management for Optimizing Media-Intensive SoCs

Tim Mace, Technical Marketing Manager, ARM

September 2009

Synopsis

The drive to constrain product costs and power consumption places severe limits on system designers, particularly with external memory bandwidth. To increase performance, designers need to focus on efficient use of this system bandwidth. In turn, this drives the need for products to help with the analysis of the system dynamics and intelligent fabric architectures to manage the scarce resources. This article examines the role of Verification and Performance Exploration (VPE) and introduces the design objectives behind the advanced Quality-of-Service (QoS) mechanisms to optimize the performance that can be delivered by such systems.

Introduction

Customers now expect a user experience whether in the home or on the move just like they have on their PCs and laptops - but better. This is driving the need for dedicated graphics and video hardware in next generation SoC designs to support an intuitive, more graphically enhanced user interface and personal entertainment experience. Furthermore consumers expect to download applications and access "cloud" based services with ease and more quickly than on the PC. In short ARM's semiconductor customers are being asked to create high performance and lower power general purpose computing platforms. For the record: fans, big battery packs and climate warming power consumption are not allowed! The product designer must now respond by squeezing out every last drop of performance from their system design. This requires a much greater understanding of the way that the software and hardware system behave under a wider range of consumer use cases and also requires a level of control to optimally partition the system resources as the system is stretched to its limits.

This has driven continued interest in a range of areas including:

- IP efficiency, particularly with the dynamic memory controller
- Traffic profiling and performance exploration
- Quality-of-Service at both the transaction level and the system level

Memory Controller Efficiency

New standards in dynamic memory generally follow two stages of introduction. The first stage is driven by the requirement to improve cost per bit and bandwidth, the typical target for the latter being twice the previous generation. The second stage is to replicate the previous generation's performance but at significantly lower power. These low power memories are referred to as Low Power (LP) or Mobile DDR. Today's low power memory technology is LPDDR (formerly MobileDDR) but the following generation, LPDDR2, is close to being commercially available. The power constraints through the form factor and the desire to support 'all day' computing restrict designers of these products to using this memory technology.

The maximum available bandwidth available from LPDDR2 devices is often limited by the power budget. To maximize the bandwidth available to the system requires a memory controller that operates with as high efficiency as possible, ideally limited only by the characteristics of the memory device itself.

Performance Exploration

With power and cost limitations for the memory system, the system architect cannot offer the guarantees of bandwidth and latency to each processing element in the system by over engineering the product. This bandwidth and latency will depend on the interconnect traffic from the other processing elements in the system. A key part of the verification process must therefore consider the variations in traffic profiles from all processing elements. This significantly increases the verification task.

For accuracy, the system designer may wish to exercise the system by running a variety of code and data sets on RTL models of the processing elements in the system, however the simulation time is too long to be practical. Options available include emulation technology or FPGA implementation, but the iteration time as the system designer changes the design topology can limit the exploration of the design. Cycle accurate models using SystemC, if available, can address the design iteration time and, to some extent, the simulation time but the system designer may not know what applications will be running on the product and may not have any code available. In any case, to generate the range of workloads on the system that are needed to explore the system performance would require a suite of applications.

However, the objective is to explore the performance of each of the processing elements in the presence of traffic from the others in the system. This does not require that the interconnect traffic is cycle accurate for a particular application. In fact, the objective is to see how the particular processing element behaves in the presence of other traffic in the system and this traffic level needs to be varied across a range of loads.

When the performance needed from the system approaches the capacity of the system it is insufficient to make some safe assumptions about the efficiency of the dynamic memory controller in the system. The efficiency will be strongly dependent on the characteristics of the traffic arriving at the memory controller including:

- Burst length
- Address distribution
- Read-write distribution
- Throughput

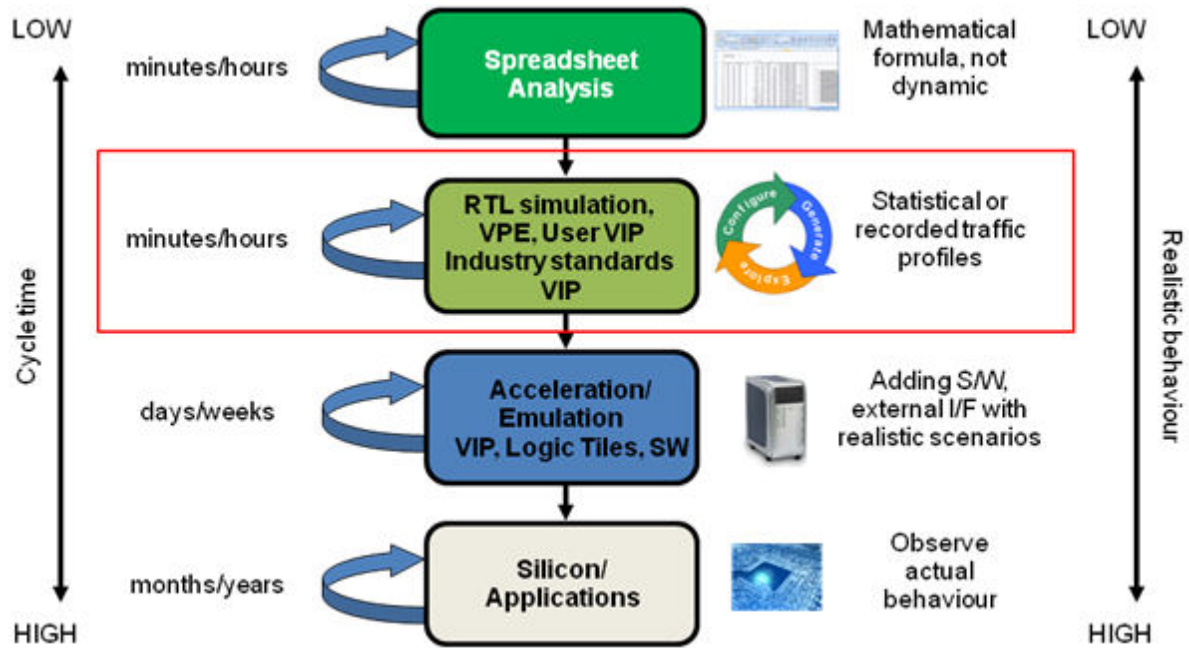
Therefore, the traffic profiles generated to represent the other processing elements in the system should share the same, or similar, characteristics. These characteristics are almost entirely determined by the characteristics of the processing element and the code being executed on it.

The VPE product provides the capabilities to support this level of system analysis. This powerful component is built around a traffic profile, which is a small binary file containing the statistical distributions of the range of fields within an AMBA® AXI transaction as well as the intra- & inter-transaction timings. A graphical user environment allows the designer to tailor this profile to match the characteristics of the represented processing element. The traffic profile can be created from the designer's knowledge of the behavior of the system component. For simple components, where the traffic is highly structured, this may be the most efficient mechanism. However, for more complex components like a Mali® GPU or Cortex® CPU, a more effective mechanism is required. A VPE monitor records the AXI traffic behavior and performance data from an RTL simulation and extracts the statistical properties directly into the traffic profile file.

The VPE master and slave components generate interconnect transaction requests and responses from the traffic profile and then execute them at the cycle level through a SystemVerilog simulator. These transactions include parameters for all relevant timings and payload information. The generation of the timing and payload content of the transaction lends itself particularly well to random generation techniques, based on a statistical profile. Random generation has been shown to be a very effective verification and validation methodology being less prone to systemic gaps.

VPE provides the flexibility to enable the system architect to explore the design space. By limiting the behavior to the transactions on the interconnect, the simulation performance of the VPE traffic generators is dramatically

faster than the RTL of the system components that it is representing. Simulations with VPE complete in minutes compared to equivalent RTL simulations that take hours.



System Traffic Management

Ideally all of the memory required by the processing elements in the system would be available locally as on-chip memory but the density and cost of this makes it prohibitive. Consequently systems use low cost external memory accessed through a memory controller. Power and system cost constraints limit the bandwidth available from this memory. The transactions in the system are dominated by the access to this external memory and the role of the system communication infrastructure is to partition this resource optimally between the processing elements.

Each of the system masters needs to receive specific characteristics from the memory system to perform its function. These characteristics can be considered as a contract between the master and the memory system. However, these characteristics and, consequently, the contract will vary from one bus master to another.

Some masters are self regulated, requiring a specific bandwidth with constraints on the latency of the transactions. A good example of this type of component is an LCD controller, where the bandwidth is determined by the pixel rate to the display. Typically this component will support a FIFO buffer to provide some flexibility to accommodate the variations in the latency of the transactions. The LCD controller will operate correctly as long as data is available when required. This implies a contract to the system of a maximum latency. If the component receives a lower latency to its transactions there is no benefit to the performance of the component but transaction latencies greater than the maximum will result in a failure such as a glitch on the screen.

Other types of master have real-time constraints, but operate on batches of data where the processing of the batch should complete within some specified time period. A graphics processing unit matches these requirements; it needs to compute the pixels for a frame before the frame is displayed. The latency for any

specific transaction is not critical but, over time, the processing rate must be sufficient to meet its goal. While it may not be possible for the system to know how far the component is through the batch process it can use the transaction rate as a proxy. By benchmarking the component over a variety of tasks, it is possible to specify a minimum transaction rate as a contract with the system to ensure the desired function. Providing a higher bandwidth increases the rate at which the frame is processed but the performance of the system as seen by the user does not. Reducing the transaction bandwidth will cause the frame rate to drop with a reduction in system performance.

The operation of the Cortex[®] CPU in the system is highly dependent on the latency for each transaction through the system. The latency for a specific transaction may not be critical, but the performance of the Cortex[®] CPU will depend on the average latency seen across all of its transactions. The contract between the Cortex[®] CPU and the system will typically be to minimize the average latency for its transactions but there may be a maximum average latency necessary for an application to operate correctly.

The partitioning of the accesses to and from the slaves in the system, particularly the memory controller, is determined by the arbitration policy used in the system with the latency seen for each transaction dependent on both the efficiency of the memory controller and the number of transactions of higher priority already in the system. The arbitration policy implemented by the memory controller should try to schedule the transactions in such a way as to maximize the efficiency from the memory controller, which will reduce the average latency seen by all masters in the system. Beyond that, the arbitration policy simply determines the distribution of that latency between the transactions.

A fixed arbitration policy delivers a fixed distribution of latency. The bandwidth and latency seen by each master is dependent on the transactions generated by the other masters that will vary over time and between system use cases. To meet the contracts between the bus masters and the system, the arbitration policy must ensure that the contract is met under worst case conditions implying that, for much of the time, each bus master is getting better transaction performance than it needs. For many of the bus masters this results in no increase in system performance but requires spare capacity that is not available within the product power and cost constraints.

To address this flaw, an arbitration policy is required that attempts to meet the contract requirements of bandwidth and latency to each bus master, but makes capacity in excess of these contracts available to the rest of the system. In situations where all contracts are being met and the Cortex[®] CPU is unable to use the spare capacity at the memory controller it is valuable to let some masters, such as the Mali[®] GPU, use this spare capacity to exceed their target rate. This makes it possible to reduce that target rate later in the frame when the system requires more capacity, yet still achieving the processing of the frame within the frame period.

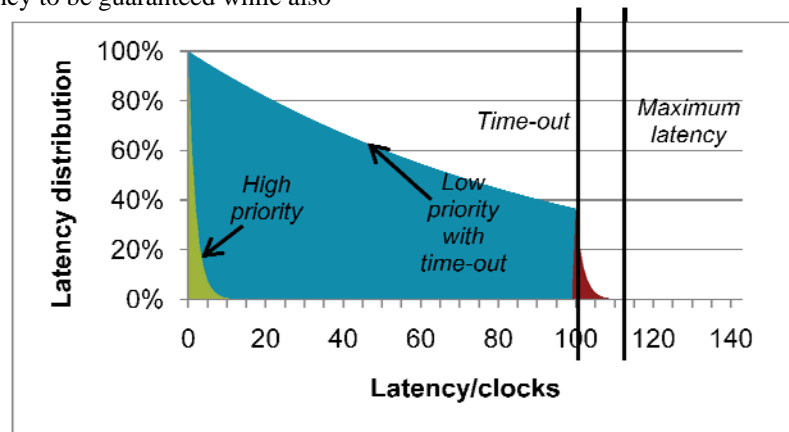
Arbitration policies exist to determine the correct order of transactions when the system requirements exceed capacity. One transaction will be more important than another, and this can be encoded as a priority. However, this priority may change over time. The AMBA[®] Network Interconnect from ARM introduces a QoS field to the transaction address to encode this priority, which crosses the interconnect with the transaction and determines the operation at each arbitration point in the system. The arbitration decision should depend on the priorities of the candidate transactions and not on the location of that arbitration point in the system. This implies that the priority encoding should transfer through to slaves such as the memory controller that implement their own arbitration function.

The operation of the Quality-of-Service (QoS) in the system is, therefore, the allocation of relative priorities to the transactions in the system. Central control of priority values is difficult to scale and maintain. This indicates that the priority values should be allocated to transactions as they enter the system. The only information available at that entry point to the system is the previous characteristics of the transactions from that master. For the system to be stable, it is necessary to avoid the ‘race to the top’, where each master increases its priority to achieve its performance, resulting in all masters having the same (high) priority value. This stability is achieved by setting the priority associated with each transaction to be as low as possible to achieve the performance

defined by that master's contract. This has the useful effect of allowing masters that can use additional capacity to do so if the contracts of all other masters are met.

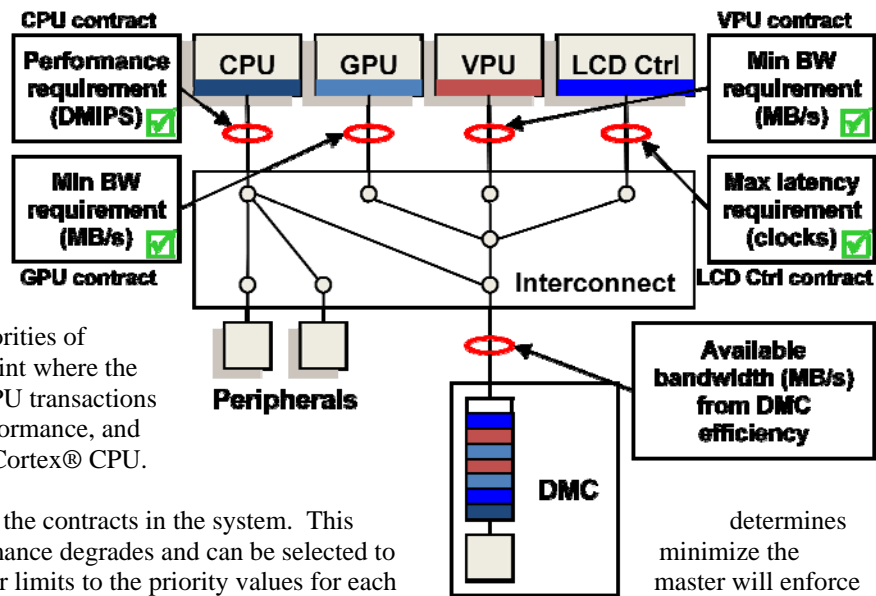
Masters such as the LCD controller can use a relatively low priority, which will increase the latency seen by the master. To guarantee the maximum latency required by their contract, the queue in the memory controller needs to count the cycles that the transaction has been waiting. If this exceeds a limit, the priority of the transaction must be promoted to the highest in the system. With the self regulated nature of the transactions from this device, this enables a maximum latency to be guaranteed while also

maximizing the average latency to each of their transactions. In turn, this reduces the average latency to the other transactions in the system. The difficulty in supporting this 'time-out' mechanism throughout the system can be addressed by prioritizing these transactions through the interconnect, but returning them to the lower priority once they enter the memory controller queue.



Masters such as the Mali® GPU have the ability to generate many outstanding transactions. The role of the interconnect is to limit the number of outstanding transactions to the system; managing the memory controller queue length while hiding the pipeline latency through the system. With a relatively constant number of outstanding transactions, the average latency seen by each transaction varies inversely with the average transaction bandwidth. A simple control loop can vary the priority to track the target bandwidth and corresponding latency.

For the Cortex® CPU, while the contracts are being met for the other masters, their priority values will be low. This gives the Cortex® CPU transactions higher priority, minimizing the latency for the Cortex® CPU and consequently maximizing its performance. As the utilization of the system increases, the priorities of other masters increase to the point where the latency seen by the Cortex® CPU transactions increases. This reduces the performance, and therefore the bandwidth to the Cortex® CPU.



There is, therefore, a priority to the contracts in the system. This is the way that the system performance degrades and can be selected to minimize impact to the user. Setting upper limits to the priority values for each contract priority. It may be that the LCD controller failure is more perceptible to the user than a reduction in Mali® GPU frame rate which, in turn, may be more perceptible than a reduction in Cortex® CPU performance.

Conclusion

Without dynamic arbitration, system architects can choose between optimizing for efficiency and guaranteeing the performance from their system. With dynamic arbitration using NIC-301 and QoS-301, system architects

can achieve both. Furthermore, analysis tools such as VPE speed up the process of finding the best configuration of these system components for the lowest power and highest performance in the tightest SoC budgets.