Multicore ARM SoCs Face Cache Coherency Dilemma

Mirit Fromovich — Cadence

Explore Cadence IP here

ARM is taking its inherently low power architecture to higher levels of performance with multicore processors.

If ARM continues to own the mobile CPU socket, quad-core and beyond processors will become the norm for high-end smartphones and tablets. But while multicore SoCs offer the promise of high performance coupled with low power, designers of these chips will also face a very tough technical challenge -- enabling hardware cache coherency.

Cache coherency ensures that each core operates on the most up-to-date data, whether it resides in its cache, another core's cache, or main memory. That sounds simple but it is actually hard when you have multiple cores working in parallel and continually updating their cache contents. In the past, software running on the cores has been used to keep cache contents up to date. However, since this is a comparatively slow activity that drains power, it is best performed in hardware.

In order to support a standard way of implementing cache coherent systems, ARM published the AXI Coherency Extensions (ACE) specification in 2011 to add to their set of AMBA protocols. ARM also released the CCI-400 interconnect IP which provides a cache coherent interconnect to link the components in a mobile SoC. It is these key building blocks, the ACE specification and CCI-400 interconnect (which tie together cores such as the A7 or A15) that enable multicore cache coherent systems. However, they also present new challengers for verification engineers who must ensure that the mobile SoCs will function correctly.

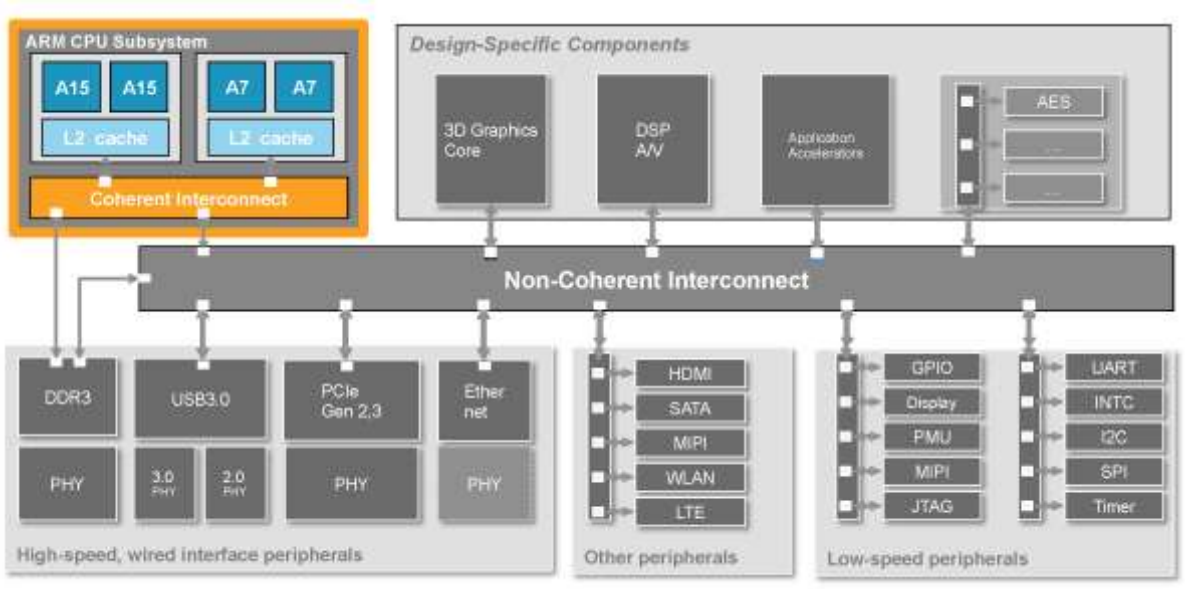

Figure 1. Example multicore SoC with coherent interconnect in processor subsystem

ACE Verification Challenges

Verifying coherency in any multicore system is hard because there are often subtle differences between legal and illegal operations. To gain an appreciation for this, let's look at an example of 2 different types of read commands in a 4 core system such as shown in figure 1. We'll name the cores in this system M1 through M4. In the first case, we'll assume that M1 issues a *ReadClean* command to retrieve a certain data item. The latest version of this data item may reside in main memory or it may be in the individual caches of M2, M3, or M4.

The interconnect needs to find the latest version and send it to M1, so the first step performed is to snoop the caches of the other cores. In this case M3 happens to have the most recent data, so the interconnect fetches that data, writes it to main memory, then sends it to M1. You can see that the interconnect "cleans up" the main memory, hence the name *ReadClean*.

Now let' s consider a second type of read operation. This time M1 issues a ReadUnique command for a certain data item. The interconnect performs the snoops and again sees that M3 has the latest version of the data. However, instead of updating main memory, the interconnect simply sends the data to M1. The main memory data item

remains out of date and M3 owns the responsibility to update it at a later time. There are nearly 20 types of read and write operations defined in the ACE specification, so constructing verification scenarios to test each type of operation is a significant challenge.

Another major challenge in verifying ACE-based systems is the tremendous size of the verification space. The number of combinations (referred to by verification engineers as the "cross products" or simply "crosses") of ACE specification elements such as Transaction types, Response Types, Domains and Caches states is huge, and every combination and permutation must be completely verified. A general rule of thumb is that the number of crosses that need to be verified with an ACE interconnect is approximately 7000 x N where N is the number of masters. So, in quad-core system, 28,000 cross products need to be verified.

Given this, it seems quite obvious why the CCI-400 (the first cache coherent interconnect developed by ARM) has only 2 full ACE masters. The challenge lies not only in the complexity of the design but also in the scope of the verification effort. Other non-coherent interconnects like NIC-400 (another ARM interconnect) can be configured with more than 100 masters and slaves, but when coherency is required, it is become impossible to provide this kind of flexibility.

Now let's look into the verification problem a little deeper. In ACE parlance, a "scenario" is a combination of the following factors: transaction type, initiating master cache status, responding master(s) cache status, and response value(s). If, for instance, the number of full ACE masters in the interconnect is 8, will 8*7000= 56,000 scenarios actually need to be verified? Well, not necessarily. Many of the crosses are not legal (for example there cannot be a situation where two masters both hold the same cached data item in the Unique state or Dirty state). So, efficient verification requires that the verification space be reduced to only to the relevant permutations.

Unfortunately, testing the logical handshaking between individual cores and the interconnect, while complicated, is only part of the verification problem. We are talking about multicore systems, and each core may be issuing commands in parallel. Sorting it all out is a very difficult task for the interconnect, and a tremendous challenge for the verification engineer who must create test scenarios that are sophisticated enough to find corner-case bugs hidden in the system.

Another complicating factor is that ACE supports the concept of Barrier transactions. Barriers cause a halt in the processing flow until some condition is met. When a master sends a barrier transaction it requires that commands initiated before the barrier was issued are allowed to complete before the barrier takes effect. Again, with multiple cores potentially issuing barrier transactions, the burden on the interconnect (and verification engineer) multiplies.

Everything we've discussed till now pertains only to the processor subsystem containing the cores (eg A7, A15) and the coherent interconnect (e.g. CCI-400). This processor subsystem is connected to the rest of the SoC through a non-coherent interconnect (e.g. ARMs NIC-400). So, verification scenarios that operate through the non-coherent interconnect should be run in parallel with the processor subsystem tests to search out bugs that only occur when the full chip is operational.

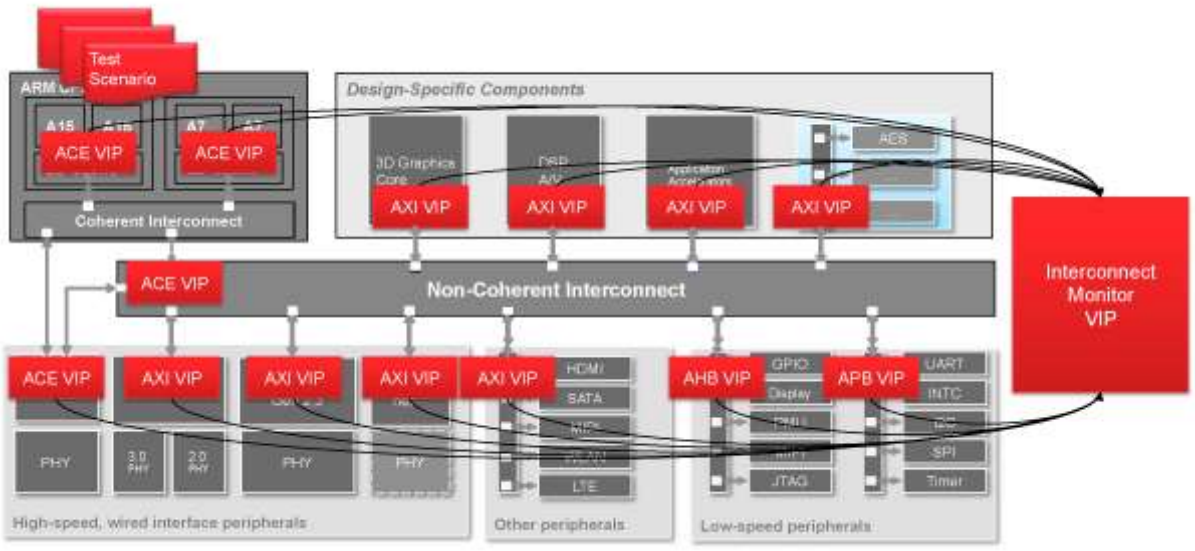

Figure 2. Example SoC showing type and placement of VIP components

Verification IP Requirements

Given the difficult verification challenges described above, it is critical to somehow offload part of the verification effort. Just as IP components offload the design effort, verification IP (VIP) components offload the verification effort. The type and placement of VIP components that are required to verify the type of SoC we have been considering are shown in figure 2. To be effective, VIP must deliver three major capabilities necessary to verify ACE-based designs. They are:

- 1) Mimicking all possible scenarios to cover the full verification space
- 2) Ensuring coherency and system compliance with the ACE specification
- 3) Measuring coverage and ensuring verification completeness.

1) Stimulus Generation

The complexity of verifying ACE-based designs requires the use of Verification IP that understands the protocol behavior of the various types of ACE masters and slaves in the system. This offloads the user from having to know the protocol details required to create legal (or illegal) transactions.

VIP that purports to mimic processor behavior must create stimulus that takes into account the protocol rules, includes a cache model, and incorporates any design-specific constraints when generating transactions. The VIP must also calculate timing correctly according to the bus state.

2) Coherency Checking

Each master and slave must be verified individually to ensure it complies with the specification, but this is not enough. The VIP must be teamed with an interconnect monitor that watches all the traffic on the interconnect. This is necessary to ensure coherency of the full system. This means the ACE and interconnect monitor VIP are required to perform two key tasks:

- a. Ensure that each individual component (e.g processors, memory) behaves correctly.
- b. Monitor the interconnect to ensure that communication between all blocks is accurate and in compliance with the ACE specification.

Item (a) is enabled by the VIP's master and slave agents. Once the user has integrated the block RTL with the interconnect, the VIP will be used as a passive agent so it must monitor each individual bus interface to ensure protocol compliance.

In order to enable item (b) a separate interconnect monitor is also required. Without such an interconnect monitor it is impossible to ensure full system coherency. This monitor must check data integrity and correctness of the interconnect itself to be sure it is behaving in compliance with the ACE specification.

For example, only the interconnect monitor, which is aware of all the masters and domains in the design, can check whether a coherent transaction initiated by a master causes the interconnect to create snoop transactions in the correct domain and master.

3) Coverage

As previously mentioned, the huge state space associated with the ACE based designs presents a key verification challenge. Simply defining all the complex scenarios requires major investment in and of itself. Yet, this is not sufficient. The ACE verification solution must enable you to also measure and ensure completeness of the verification space.

A coverage model is used as the metric for determining verification completeness. Therefore it is imperative that the VIP provide the complete coverage map based on the ACE specification. As mentioned above, performing

verification efficiently requires reducing the coverage space to only monitor legal cross products. The coverage model serves this function.

Learning More

Cadence will be presenting a detailed analysis of the challenges outlined above and a full description of its ACE and interconnect VIP solution at ARM TechCon in session ATC-106.

If you can't attend the conference, we'd be happy to mail you a copy of the paper. Please complete this protocol interest survey and we'll mail it to you after the conference.