Hierarchical IP Subsystems

IP Usage in SoCs

Most practical SoCs are more than just a collection of IPs and custom development. There is a clear hierarchy associated with the IPs that are used. For example, an SoC might instantiate 8 USB ports, but each port needs a USB phy and a USB controller. Further, this SoC might also need to instantiate other IO like PCIe or SATA etc. All those individual functions require their own controllers, PHYs etc.

Putting all of these together into a proper hierarchy is an early requirement.

Selecting individual IPs

As SoCs become more complex, it becomes harder to select the IPs that are needed to build the system. Many decision points need to be considered - does a specific release of a USB controller work with a specific release of the USB PHY? Will there be any issues integrating this bundle into the context of the chip? Do we have permissions to use these specific versions of these IPs?

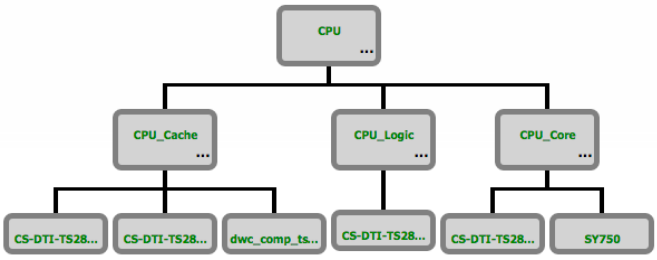

Fig. 1: Example of the hierarchy in a "CPU" subsystem.

Working with Subsystems

Several of these issues can be resolved if there's an easy way to build subsystems. An IP management system like ProjectIC that understands hierarchical designs can be of significant help.

ProjectIC has been built from ground-up to handle hierarchical IP subsystems. Each "IP" in ProjectIC can be a standalone design object, or a complete hierarchy. Thus, users can include IPs in their project at any level that makes sense.

Advantages of Using Hierarchical IP Subsystems

There are several advantages of hierarchical IPs:

1. **Abstraction**
   It is quite easy to integrate fully capable functional blocks in your design rather than a laundry list of individual IPs. This is much quicker and more intuitive.

2. **Compatibility**
   Does version 10 of the USB controller really work with version 7 of the PHY? Should I use the latest version of both these blocks and hope for the best? These kinds of questions can be eliminated by simply choosing the USB subsystems, where proven configurations with versions that work with each other.

3. **Dependency Management**
   Bringing in a subsystem automatically brings in all the dependencies that are needed. There is no need for cumbersome dependency discovery, since the subsystem will bring in all the needed dependencies.

4. **Discovery**
   By looking at the hierarchies in which a particular IP of interest is used can help with easier discovery of the various components available to the team. For example, if the USB PHY is used in a hierarchy that contains other IO interface blocks, it is very useful to be able to discover the context in which this block is frequently used. This can aid the design process immensely.

Support for Hierarchical IPs

ProjectIC has many features that support hierarchical IPs.

Every IP can have a list of 'resources' - other IPs that are in the system. These resources can also have resources of their own, and so on. Building a hierarchical IP in ProjectIC is as simple as including another IP as a resource. The tool automatically figures out any other resources that are implied by including this IP.

Building a workspace with a hierarchical IP is handled seamlessly. All the resources that are part of the hierarchy are automatically instantiated in the workspace.

Resource locations inside a workspace can be controlled with fine granularity.

The tool automatically checks for circular dependency between resources, resource conflicts etc.

Container IPs can be used to create hierarchy levels and compatible bundles.

Each resource in a hierarchy can be independently moved from one version to another. This can be done in multiple different ways - from the command line, from the GUI etc. - allowing for maximum flexibility in interacting with resources.

Several other features like hierarchical releases, property inheritance, IP hierarchy traversal, tree views etc. are also fully supported.

Conclusion

In any practical SoC design, the ability to discover, use and interact with hierarchical subsystems can make a huge difference in the design process and IP reuse. IP management tools like ProjectIC that natively support hierarchical IPs can be leveraged for this mode of IP reuse.

For more information on Hierarchical IPs, please email contact@methodics.com.