Best Practices for Maximizing IP Reuse in SoC, IC and FPGA Design

By Alex Tumanov, PhD.
IC Manage, Inc.

OVERVIEW

The ability to create differentiated products for SoC, IC, and FPGA designs within narrow market windows depends heavily on how effectively design IP is reused in the design process.

This white paper covers best practices for maximizing the efficient reuse of internal and external IP from a data and design flow integration perspective. It discusses core development practices which lay the groundwork for quality IP development, along with efficient mechanisms for managing a company's dynamically changing IP across divergent design teams and derivative designs.

IP Reuse best practices commence with the partitioning of new design work into both logical and functional modules suitable for later reuse. One key component is establishing a change management and IP delivery methodology which utilizes virtual copies of the IP; this step enables management of the interdependencies between the various IP versions in use across multiple chips and design teams. This virtual copy approach replaces ad-hoc copying of IP data around the enterprise.

This white paper also discusses the productivity advantages of linking IP repositories with bug tracking systems to enable bug tracing between the original IP and all its versions. This step allows for automated bug notification to chip leads and IP owners to avoid taping out known design errors.

Further efficiencies can be obtained by establishing an automated check-list driven IP development process and acceptance criteria. Other IP logistics management practices include instituting security protocols to reduce the risk of theft from both internal and external sites.

SETTING UP IP DEVELOPMENT TO MAXIMIZE LATER IP REUSE

Partition new design work into functional modules.

The project manager or chip lead creates an initial partitioning structure, and then assigns the design modules/IP development to individual designers or teams. This practice establishes the various design modules as IP blocks, even when their current reuse potential is not yet known.

The ideal practice is to define the design modules in anticipation of the optimal reuse of each block. The lead must strike a balance between creating small modules, which have high flexibility as they will not be further split during reuse, and ensuring that each IP block contains a critical mass of functionality, so it can be reused on its own under certain conditions rather than always needing to be combined with other modules. By encouraging only selected team leaders and architects to define the individual blocks and their components, development organizations can avoid complex instance trees and broken rules.

Organize each IP block according to data type.

Each IP design element is generally comprised of mixed data types, including HDL, SPI, LIB, LEF/DEF, SPF, and GDS. Setting up the IP block organization by data type gives developers an infrastructure in which to place their data files and sub-directories rather than using ad-hoc trees which may be difficult to package or release. Data type partitioning also allows engineers to easily extract specified data types, such as "Verilog and LIB only".

Put all new IP – internal and external - into the IP repository.

Use scripts or other automated mechanisms to check in, map, or import all IP - both new and legacy - according to the partitioning, data type, and repository structures discussed above.

Decouple the repository name space from the actual disk hierarchy, so that disk structures can be directly remapped. This practice avoids having to change the repository structure, yet allows for re-factoring of the hierarchy without breaking scripts and flows.

Entering 3rd party IP into the same repository structure allows easier integration of the new IP into existing design flows to track and verify any changes associated with updates from external vendors.

Link the IP repository to a bug tracking system from the start.

Integrate the IP repository to one or more bug tracking systems - if a bug tracker is not already embedded in the IP logistics management system. Doing this step from the beginning ensures that all identified bugs and bug fixes associated with each IP block are automatically recorded in both the bug tracking system and the IP repository.

This close linkage allows design and verification teams to view and trace the bug history for every IP they work with across all versions and designs. The IP changes associated with the bug fixes should be viewable in the bug tracking system, and the bug history viewed in the IP repository. Further, chip and IP designers should automatically be notified of any new bugs identified as well as the bug status. Bug fixes can then be automatically or selectively propagated to the various versions.

This practice allows the chip integration lead to mark the IP blocks as 'finished', then decide whether to let them stay finished after reviewing the defects for a particular release. This methodology allows the mix and match of different versions of the same IP in different blocks, without having to respin all blocks to use the newest version.

Use an assembly methodology to capture IP dependencies in a design.

Use structured techniques to define assembly relationships between various IP and hierarchical IP sub-modules. Configuring these assembly relationships in advance allows for accurate, real-time tracking and reporting of the where, what and who of each IP version as it is used across the enterprise.

Defining assembly relationships in advance ensures that every item in the assembled IP configuration is tracked in the design system from the point of creation or import.

A defined assembly approach is in contrast to manually creating after-the-fact linkages or a 'bill of materials' (BOM). Since the BOM and associated records are typically not created until the end of the project, early visibility into project dependencies are lost.

Finally, a defined configuration approach enables the ability to deliver bug roll up reports for the assembled design without having to copy the bug report into each chip project. The heavy lifting can be achieved automatically using the assembly map rather than scripting and maintaining a complex methodology or reverse tracking through a static BOM.

EFFECTIVE USE OF BRANCHING

While creating new IP derivatives, use virtual referencing techniques instead of copying the IP.

Clone the IP to create virtual copies through pointers, so that each IP version can be separately and independently tracked. This practice replaces creating physical copies of different IP versions in multiple repositories, leading to unrecognized and orphan copies whose usage is unknown. Maintaining the relationships between the original IP and its derivative versions allows for the efficient management and propagation of IP updates in both directions - parent to child and child to parent - depending on the policy.

Advanced branching with history tracking allows top level assembly teams to work with stable or released versions of each IP module while the design teams continue to work on the IP and chip development. By avoiding data duplication, branching also minimizes storage space in the repository.

Use private branching to manage IP block derivatives within the same design.

Encourage designers and teams to use branching for development on their individual IP modules, and then merge their changes once their tasks are completed. When using different versions of the same IP module multiple times in the same design, it is best to create re-named branches for each IP version to ensure accurate tracking, and to avoid name space collisions. The renamed branches still carry the original history since they are virtual projections rather than physical clones.

DEVELOPING QUALITY IP FOR REUSE

Establish the practice of continuous integration.

It is a best practice to merge new or changed IP code into the IP repository with sufficient frequency that developers collaborating on the design or IP can notice errors and begin correcting them immediately. The designer's changes should be submitted as a single commit operation using a repository that supports atomic change list creation, and it should be easy for all team members to view the latest deliverables and changes.

Continuous integration forces ongoing team member communication, reduces the number of conflicts, and accelerates resolution. In contrast, delayed or only periodically-scheduled check-ins can make it more difficult to resolve development conflicts.

Set up and utilize a checklist-driven flow during IP development.

Encapsulate all relevant property data associated with each IP element into the IP repository, such as design data, bug status, assertions, constraints, timing goal status, margins, power consumption, electrical data, regressive results, and documentation.

As the IP development and verification process progresses, members of the design team involved with the IP can mark or flag its status, based on rules or metrics associated with the property list. Quality or verification metrics can also be imported or linked from third party tools. In combination, these steps allow managers to measure progress against specific milestones such that certain property groups go from "incomplete" to "in progress" to "pass".

Optimize security access for IP blocks.

It is important to put security measures in place to control user action and for file access control for the IP block to protect the IP data and restrict who can use certain blocks. The security granularity can range from chip level to IP block level down to the individual file.

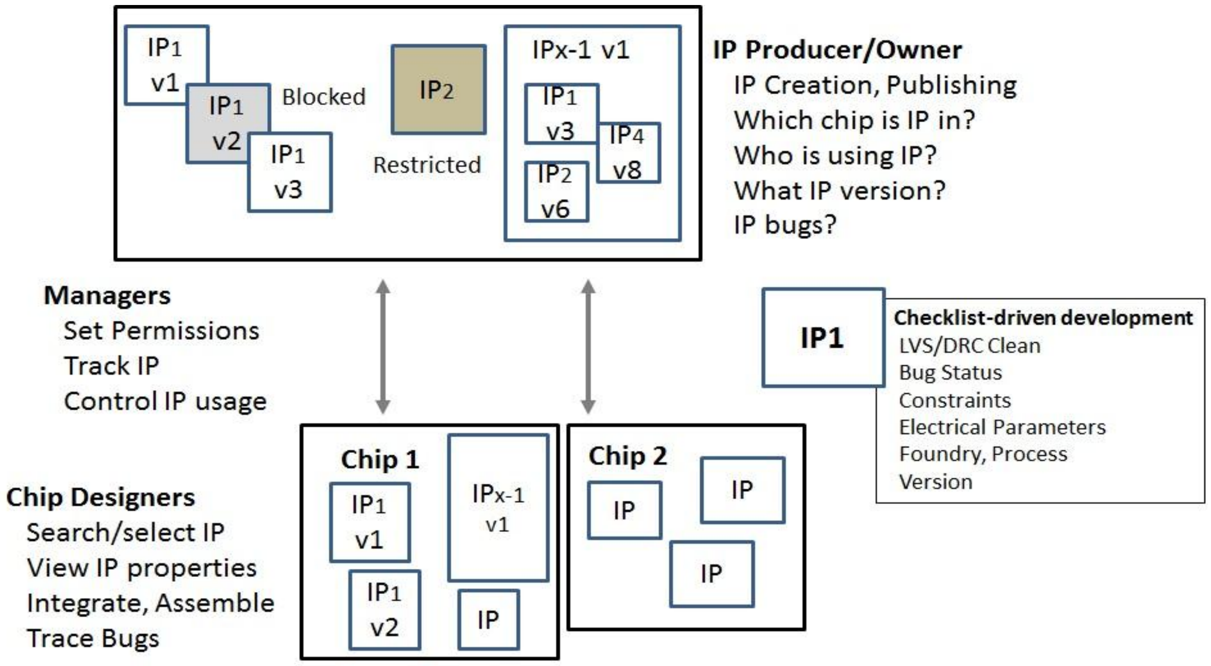
In setting up the security structure, the organization's security needs must be balanced with the development restrictions and overhead associated with maintaining complex security policies. One practical approach is to set general policies using a fairly high granularity, then modify and tighten the measures for specific situations.

CONCLUSION

By establishing and deploying best practices for IP creation and IP logistics management, organizations can highly leverage their company's development assets across the globe.

As seen in the figure below, today's designs have complex interdependencies to navigate, across the original IP, its versions, and even the individual hierarchical sub-modules for larger IP blocks. For example it is critical to enable bug status visibility and tracing between the original IP and all its versions.

The design practices discussed above can form the preliminary basis for a company's IP reuse and logistics strategy; they can be further adapted and extended to match the needs of individual organizations. Dramatically reducing the engineering time spent integrating internal and third party IP into the design flow, and minimizing the time to achieve confident verification of the IP in the context of the entire SoC, IC, and FPGA design, will enable broad compliance by worldwide design and verification teams.



IP Reuse Interdependencies in Design

About IC Manage IP Central

IC Manage IP Central is a platform for maximizing IP reuse; it manages IP interdependencies and automates IP logistics across the enterprise. Design and verification teams can use IP Central to rapidly publish and integrate IP into their existing design flows, and to trace bug dependencies. Internal and third party IP can be imported or linked to IP Central from multiple commercial and open source design management and internal revision control systems. IP Central enables close collaboration between IP owners, chip designers and chip leads.

About IC Manage

IC Manage, Inc. provides high performance design management solutions for companies to efficiently collaborate on single and multi-site designs and obtain maximum IP reuse. The IC Manage Global Design Platform (GDP) lets designers securely track and distribute design, configuration, dependency, and IP property data on multiple projects across the globe. IC Manage's IP Central is an IP logistics platform for integrating, publishing, and managing IP interdependencies. IC Manage is headquartered at Suite 100, 15729 Los Gatos Blvd., Los Gatos, CA. For more information visit us at www.icmanage.com.

Alex Tumanov, PhD, Manager, Applications Engineering, IC Manage

Alex manages applications engineering at IC Manage, where he has supported IC Manage's customers in structuring and implementing their IP and design management systems. Prior to IC Manage, Alex worked as management consultant at McKinsey and Company. Alex was also a researcher at Rice University and Stanford University, where he developed an object-oriented data acquisition software system to detect, simulate, and analyze billions of interactions, built a particle detector processor using programmable logic Xilinx chips, and developed modules to analyze billions of interactions. Alex received his M.S. and Ph.D. degrees from Princeton University.