Arasan - Incorporating Quality into Reusable Interface IP

By
Somnath Viswanath – Product Marketing Manager

November, 2009

Overview

Today's complex silicon-on-chip (SoC) designs contain multiple instances of silicon intellectual property including CPUs, DSPs, and large numbers of interface IP—SD, SDIO, USB, and MIPI—to store and route video, audio, and data within these designs. On some large SoCs, as much as 85 percent of silicon real estate is made up of third party IP. Given the time-to-market constraints, chips specially destined for consumer electronic devices, the standardization of individual interface IP blocks has made it very attractive to do a make versus buy decision. This has created a large and steady demand for interface IP from third party suppliers, however the expectations of quality for first time success are extremely high.

Beyond shortening time to market, the advantage third party IP provides SoC designers is reducing the design resources needed to build a large system on chip while mitigating risk. To deliver these benefits, IP blocks must be as close to plug-and-play as technically possible, considering the variability in complex SoC designs. The worst-case scenario for a customer purchasing an IP block with an error is having to divert resources from the larger SoC effort to help find the problem and even worse being delayed in getting the chip into silicon. Hence pre-silicon verification is extremely important to ensure against a costly respin of the SoC. Silicon proven Interface IP blocks are less likely to cause this problem than those with little or no history. Successive generations of a given interface IP block builds on existing extensively debugged RTL code, hence maintaining the quality is desired.

The key element of a successful IP block is being able to seamlessly integrate into a larger design, especially with the myriad of internal buses. Seamless integration demands that the interface IP deliverables come with support that understands the SoC's operation in the final consumer application. For example, the support team should have a comprehensive understanding of the USB, SD/SDIO, or MIPI specification as well as internal buses like AXI, AHB, APB, PCIe, PCI etc and how the IP block will behave within many different SoC environments—cell phone, netbook, video game, digital camera, and the many other consumer products being developed. In addition, the IP blocks must be customizable and verifiable to accommodate the unique requirements of individual customers.

Introduction

Arasan Chip Systems has been at the forefront of the IP business for the last fifteen years, striving to enable technology adoption by providing total solutions for standards-based peripheral IP. Its major advantage is the company's close participation with the standard bodies creating the peripheral specifications. The company has participated in the definition of the standard for every peripheral IP block it produces. In addition, one of the biggest value-add Arasan offers is providing customization services to integrate the IP into the customers SoC design. This service extends to ensuring certification and interoperability by participating in

industry plugfests for PCI, PCIe, USB, Ethernet, MIPI, SD/eMMC; testing with key partners; and testing in commercially available products—PDAs, mobile phones, consumer products.

At Arasan building quality into an IP design from inception begins with a design methodology that targets the root cause of errors: poorly or incorrectly written RTL code, errors that creep into a design during synthesis, and errors in the design's corner cases. Eliminating these errors is essential to achieving a high level of quality. In addition, complex interface IP block deliverables include a large set of design components: requirements, specifications, HDL code, compiled code, embedded software, test plans, test scripts, test results, defect logs, etc. Because the hardware RTL and its associated software drivers are develop concurrently, effective collaboration and communication among these engineering teams, often separated by time and geography, is essential to parallel design and verification. To ensure that team members have easy access to the right versions of code and documentation, a common configuration and change management system is essential.

Ensuring the quality of reusable IP begins with a tool flow comprising a well-developed verification strategy, one that roots out errors that creep into a design during development. The strategy must accommodate customizations provided for individual customers after the IP block has been productized. The verification strategy also demands a well-defined process for tracking and documenting errors identified and fixed, both before and after an IP block is released for licensing. Finally, the nature of reusable IP blocks being modified leads to a proliferation of versions of every IP block developed—a version for each customer, thus leading to continually increasing number of SKU (stock-keeping units) that must be maintained and updated over time. Arasan Chips Systems has confronted these issues head on to ensure the highest level of quality for the IP blocks it offers. This white paper will detail the process the company adopted to address all these issues.

Designing Quality in Reusable IP

Arasan has adopted a design flow with an aggressive verification regimen that includes conventional simulation, rules checkers, and formal verification tools to predict and eliminate problems that normally only show up much later in the design process. Arasan has adopted SpyGlass, a rules-based checker from Atrenta Inc. that allows design managers to capture design-and-reuse policies, thus correcting poorly or improperly written RTL code and finding errors that creep into a design during synthesis. Arasan is adopting the Cadence Encounter Conformal Equivalence Checker to assure code changes or synthesis runs do not alter the logic function. To find difficult-to-isolate errors in corner cases that other tools are unable to locate Arasan uses the Incisive Formal Verification tool from Cadence and the Assertions facility in SystemVerilog to boost verification coverage over 90 percent.

In the simplistic approach to verification, the design team produces RTL code and writes a test bench that applies stimulus vectors to the code and checks the resulting responses against expected ones. The problem in this approach is that much of the design is left untested, on average less than half the code. To boost coverage, design teams run the code through register-transfer-level (RTL) rule checker, such as SpyGlass, which employ "rule decks" that can be written in C or Perl. These decks allow design managers to capture design-and-reuse policies by adding rules without modifying the rule checker. In operation, SpyGlass checks for style limitations, makes sure that the design is using the correct naming conventions, and that the design meets certain norms. For example, when moving a signal from one clock domain to another, does the design adhere to the best practices method? Other checks might include analyzing problems such as latch inference or combinational loops.

A design undergoes numerous iterations before it is ready for backend layout and each step in this process can introduce logical inconsistencies. Arasan employs Cadence Encounter Conformal Equivalence Checker to ferret out these differences. The Logical equivalence checker (LEC) ensures the correctness of the implementation flow, verifies logic changes and enables comparison of different views and abstraction levels. LEC checks the functional equivalence of different versions of a design at each step and enables designers to identify and correct errors as soon as they are detected. For example, when a Verilog RTL description is synthesized into a gate level netlist, are the two descriptions equivalent? In combination with rules checkers, LEC can increase the equivalent coverage from 50 percent to 65 to 70 percent.

To find the errors that are the most difficult to locate, Arasan is collaborating with Cadence on delivering functional verification best practices. Under the agreement, Arasan will provide to Cadence its MIPI design IP, which Cadence will engineer into its Incisive Verification Kit, integrating their own MIPI based verification IP, example flows, user workshops and documented best practices. The collaboration will start by enabling potential MIPI users a chance to see first hand a comprehensive working environment with hands on workshops and labs demonstrating the industry leading Metric Driven Verification (MDV) methodology.

Formal Verification imposes the discipline of describing an IP design in a formal language, no mean feat as it is often as difficult as creating the design itself. Formal verification identifies many critical design problems such as dead code, uninitialized memory, bus contention, floating buses, and x value propagation. It can also verify good design practices, when using case statements, by analyzing and reporting full and parallel case pragma violations. Incisive reads the language description in Property Specification Language (PSL) as well as the RTL description and the tool produces an output that describes if the two descriptions match: equivalence checking at the highest level.

The strength of PSL is that it describes a function unambiguously. The problem with specifications that describe interface standards such as USB, MIPI, among others is that the documentation attempts to be exhaustive and thus tolerates some amount of ambiguity. The standard is also an English description that must be translated into a formal PSL lanuage. In

addition to ambiguity, I/O specs provide the designer the freedom to make the standard implementation-dependent. This cannot be captured in a formal verification description. Thus, the formal description created by one design group for a USB interface may not be equivalent to the formal description created by another group.

To supplement formal verification, one more tool that Arasan is about to embrace assertion-based verification, which goes hand in hand with the formal methodology. Designers capture specific design intent in the form of an assertion description.

The design then uses simulation or formal verification of these assertions to verify that the design correctly implements that intent.

The newest SystemVerilog contains an integrated a set of constructs that help designers develop assertions and closely couple them into the main body of the RTL design or verification code. A SystemVerilog assertion construct is actually part of the RTL itself. Thus an assertion can reside inline with other language constructs without any need to create special pragmas or similar restrictions.

What is an assertion? It is a formalized description of a function, for example, the condition, "when signal A and B are 1, then signal C must be 0." If it's not then there is an error. However, the uniqueness of an assertion is that conditions may encompass the element of time, for example if 100 clock cycles ago signal A was 1 and 50 clock cycles ago B was 1, then C must now be 0. Note that all sequence operations are synchronous to a clock.

The other great benefit of assertion-based verification is coverage, which refers to the collection of statistics based on sampling events within the simulation. This differs from the notion of code coverage that instruments the design RTL to determine that all lines of code have been executed. Assertion coverage ensures that not only is the design working according to the design specification, it determines that all desired corner cases in the design space have been explored.

Addressing Error Reporting and Tracking

In Arasan engineering methodology once the design team completes the RTL for an IP block, it sends the complete set of files to the verification team, which begins the exhaustive process of testing the RTL for code coverage, design functionality, and finally corner cases. Keeping track of and recording errors detected during the design and verification of IP is another critical element needed to design quality into an engineering methodology. This is one aspect of a design flow where a tool can codify and enforce good engineering practice. That tool is Bugzilla. A web-based general-purpose error tracker and testing tool Bugzilla is a product of the software development discipline. It was originally developed and used in the Mozilla project, and is still licensed under a Mozilla Public License. Today, hardware design teams have adopted Bugzilla as a defect tracker for HDL language programming.

Bugzilla increases productivity, improves communication, cuts downtime, and ultimately raises customer satisfaction. The tool can also help reduce costs by providing support accountability, telephone support knowledge bases, and by keeping tabs on unusual system or software issues. The tool allows individual or groups of developers to perform functions such as

- Track errors and code changes
- Communicate with other team members
- Submit and review patches
- Manage quality assurance (QA)

When the team finds an error it is recorded in Bugzilla, which automatically sends mail to the design team, which opens a ticket and assigns someone on the team to fix the error. When the fix has been accomplished, the new version is sent to the verification team to ensure the repair actually resolved the problem. If not the error returns to the open state and the process repeats. Otherwise the repair is resolved. Bugzilla is imposing a discipline on how design and verification teams interact.

Addressing configuration-version control

When an IP block is released for commercial licensing, the files comprising the IP are loaded onto a server and customers are provided a key and directed to FTP the files from the server. The simple task of hosting these files on a server presents a number of problems. The server has become a repository for files for different IP blocks as well as different versions of the same IP block—one version for customer A and a second slightly-modified version for customer B. The scenario cries out for a configuration and version control system that keeps track of the files for each unique IP block, who accessed and changed them, which versions of which files have been distributed, and who has received them.

Arasan solved this problem with CVS (Concurrent Versions System), a public domain version control system software package that has been deployed for two decades. CVS is a production quality system in wide use around the world initially deployed for large software development project, but equally and easily applicable to SoC designs written in the RTL hardware description language Verilog. Along with revision control CVS provides configuration management capability. The tool manages the configuration of files that belongs to an individual customer's version of an IP block. It keeps track of each file's unique version number, which changes each time it has been checked out, revised, and returned to the storage facility. This configuration is preserved to access previous releases once the IP has been in the field for a period of time.

A similar set of issues exists during the design, verification, and validation process before the IP block is released, which CVS addresses through a formal revision control mechanism. Each time a design team works on a project, they check out the design from a storage facility. As the team makes changes to files in the design and check the files back into the storage facility, CVS assigns the files new revision numbers to reflect the changes that have been made. The tool records the history of sources files and documents, thus enabling designers to access and modify earlier versions of the design. Arasan has also developed scripts to log CVS operations and enforce site-specific polices.

CVS enables developers separated by geography to operate as a single team. The version history is stored on a single central server and geographically dispersed clients have a copy of all the files that developers are changing. CVS provides a flexible modules database that provides a symbolic mapping of names to components of a larger software distribution. It applies names to collections of directories and files. A single command can manipulate the entire collection.

CVS eases the task of managing versions of files by allowing designers to refer to a file by name and not by the version number, thus ensures designers get the latest version unless an earlier version is specifically requested.

Finally, CVS provides the revision control and configuration management for the set of documents accompany each commercial release of an IP block. This enforces a discipline on the development process by associating a set of documents with each IP release. Each document has a version number. When the IP block is shipped and another version is shipped later, there is a complete set of unique document for each version. As with the hardware and software data files, all the documents for all IP block releases are maintained in a central repository and earlier releases can be accessed and updated.

Conclusion

Quality is an intangible asset that must be engineered into every reusable IP offering. The characteristics of a high-quality IP supplier who inculcates the confidence for first time success for its customers include the following:

1. A proven track record of design wins in silicon. Arasan's 15 year track record of tier 1 semiconductor customers testifies to the company's ability to deliver a quality product and save customers time to market, cost and risk benefits. (http://www.arasan.com/customers/). Today Arasan IPs are in products like smart phones, netbooks, Personal Computers, Nintendo game consoles, Digital Cameras and many more…

2. An in depth understanding of the interface specification. Arasan has a track record of actively participating in the standards development activity (usually from the early days) for all major

interface IP products in the company's portfolio like: Memory IPs (SD, SDIO, SDXC, eMMC, CF, MS), USB, MIPI and Connectivity IPs (PCI Express, Ethernet)

3. A continuous and evolving IP block development process. Arasan has been building successive generations of interface IP offerings upon previous generations for over a decade incorporating past experience and newer standards into each new release.

4. A disciplined design regimen. Arasan has developed a design and verification methodology to ensure the highest level of confidence that each design has been thoroughly verified and all corner cases tested.

a. An error control and configuration management system. Arasan has implemented the latest tools to track and document errors detected and corrected in each design throughout the design and verification process as well as any errors encountered while customizing an IP block to individual customers.

b. A revision control and configuration management system. Arasan revision control and configuration management system ensures each IP block shipped to a customer has its unique version of data files and document and that that changes to IP block is recorded and updated throughout the life of the block.

c. A hardware/software validation environment. Arasan ensures that all its IP products have hardware platforms with associated software stacks for customers to do development or validation.

5. Enabling technology adoption through a Total IP Solutions Approach. For every IP in its portfolio Arasan provides not only the RTL IP but also the verification IP suite, Hardware Platforms for IP system development and validation, Portable software stacks and customizable design services.

This provides Arasan's customers with everything they need to succeed.

Arasan Chip Systems Inc.

2010 N. First Street, Suite 510, San Jose, CA 95131

Phone: 408-282-1600
Fax: 408-282-7800
Email: sales@arasan.com

Data Sheets Link:

http://www.arasan.com/datasheets/login.php

For a complete directory of Arasan IPs, please visit: www.arasan.com