The search for productivity in SOC (system-on-chip) design is a search for balance between abstraction and automation. Greater abstraction at a step in the design flow means fewer design elements to process. Greater automation means that each element requires less human attention. Ideally, designers could capture an abstract representation of an SOC’s intended behavior, verify that the representation describes the desired chip, and push a button to tape-out. We are not yet there.

For years, some enthusiasts have promoted high-level design languages—often dialects of C—as the path to greater abstraction. Except in a few categories of architectural elements, however, it has been difficult to move the design beyond behavioral or transaction-level representation and into the implementation flow. “High-level synthesis is still very domain-specific,” says Ken Wagner, PMC-Sierra’s vice president of engineering. Without synthesis, designers must recode the high-level version by hand into RTL (register-transfer-level) logic.

While EDA vendors were struggling with synthesis, another kind of high-level abstraction had become common in SOC design: IP (intellectual-property) reuse. You would not ordinarily think of reusing IP as a high-level design method; it’s a simple matter of expediency. The technique is so ubiquitous that many of today’s SOCs simply couldn’t exist without it (figures 1 and 2). “In our shorter designs, often 80% of the chip is composed of IP,” says Jitu Kahne, director of central engineering at AppliedMicro.

Yet this heavy reliance on IP is an attempt to raise the level of abstraction in the design. Ideally, an IP block can be a black box that the design team need never open. IP reuse allows a design team to represent a large block by its function and its I/Os rather than by its internal structure. The design team could carry this abstraction all the way through to signoff, integrating the black boxes together rather than opening them and exploring their contents.

With IP, as with high-level languages, however, life is not ideal. “IP is often a mirage,” says Wagner. “Using third-party IP takes a large effort: 30 to 50% of the work required to design the block...
Much of this effort is due to the loss of automation, and some is due to the inability to maintain the abstraction through critical phases of the design cycle. According to Kahne, 60 to 70% of AppliedMicro’s design cycles are IP integrations. Few tools, however, are available for a design flow in which selecting and integrating IP dominate.

According to Karim Arabi, senior director of engineering at Qualcomm, 80 to 90% of the IP in the company’s designs doesn’t change from one chip to the next. “But we still have to redo the whole integration process every time,” he says. “There are no tools to automate IP qualification or integration, and the ECO [engineering-change-order] process is purely manual.”

It is fair to say that IP integration dominates the SOC-design flow. Although IP reuse can greatly increase the level of abstraction, it falls short on automation. This article examines four critical activities in IP integration: determining IP behavior, constructing interfaces, filling in the infrastructure, and closing the design, comparing how designers now perform the design tasks and speculating on the opportunities for further automation.

**IP behavior**

How do designers determine the behavior of an IP block, and how do they explore the behavior of their SOCs with the blocks in place? Both questions are nontrivial if you are attempting not to look inside the black box.

A simple answer to both questions is sufficient when the IP implements a standard function, such as a USB (Universal Serial Bus) 2.0 interface. “The IEEE has done a wonderful service by standardizing so many interfaces—both external and within the die,” observes Taher Madraswala, vice president of engineering at Open-Silicon. If a block implements an IEEE standard or a de facto industry standard, little of the block’s behavior is subject to interpretation. And third-party VIP (verification IP) exists to confirm the block’s behavior.

Not all IP implements standard functions, however. “The customer must prepare to make a significant investment to select IP to meet his specifications,” PMC-Sierra’s Wagner says. Some IP packages include executable behavioral or transaction-level models, which may not accurately model the behavior of the RTL logic. Without trusted models, the architects must rely on RTL simulations or FPGA emulations and on close study of data sheets to determine precisely what the IP does. To understand the workings of the full SOC with the IP in place, the architects may have to rely on mixed-mode simulation if it is fast enough, or they may have to write their own TLMs (transaction-level models) using these sources—a labor-intensive and error-inviting undertaking. This problem brings up the opportunity for automation: a tool that would extract TLMs from RTL code.

Some important considerations in modeling the behavior of IP are that most IP blocks are configurable and some, such as CPU cores, are programmable. Configurable or programmable blocks may require a good deal of configuration or even a software project to produce a high-level model of the IP for system exploration.

**Interfacing to the blocks**

Extracting the interface definition, converting it into design requirements, and verifying that the block connects correctly to the blocks with which it interacts are all significant design tasks in cases lacking an industry standard to unambiguously define the pins on an IP block. These tasks are now
almost entirely manual.

As with understanding the behavior, the best case for understanding IP interfaces occurs when those interfaces comply with external standards. If the interface passes inspection by independent VIP, you should be able to just drop the black box into your design. Some FPGA tools do just that task: They build systems by assembling IP blocks onto a standard bus and by building a register map and driver library for use by software developers.

The Accellera standard IP-Xact takes a similar but more ambitious approach in the SOC world, attempting to code enough information about the function and interface behavior of an IP block into a standardized XML (Extensible Markup Language) form so that automated tools can identify an IP candidate based on design requirements and generate the necessary interface code to build the block into an SOC (see sidebar “Using and adopting IPXact”). A team in the research group at Philips Semiconductors, later NXP, developed automated system-building tools to work with these XML definitions, but the group appeared to have largely abandoned the work when NXP slashed its research activities and then divested its SOC-design teams.

The principle is sound, but, as always, there are worrisome details. “Unfortunately, not everyone follows naming conventions, even on standard interfaces,” says Open-Silicon’s Madraswala. So connecting the pins may require some further research and design, especially with regard to that ancient saboteur of designs: polarity. Also, even with standard interfaces, you can’t blindly trust the test bench.

“It may be difficult to fully exercise the timing of interface IP, even with in-circuit emulation, unless you can create and apply appropriate stimuli and monitors to represent your use of the block,” says PMC-Sierra’s Wagner. His point applies even if you are staying strictly within a standard. If you are doing anything beyond the standard, such as overclocking or sideband signaling, building a custom test bench is essential. Wagner goes on to counsel engineers to ensure that the bench offers both high coverage and high performance. “Ideally, there would be static and dynamic assertions,” Wagner adds.

The question of assertions is a sensitive one. More and more SOC teams are adopting assertion-based verification strategies. But an informal survey suggests that assertions are appearing in IP deliverables at a much slower rate. This absence may force the SOC team to create their own assertions for IP, which would require them to intensively mine the data sheet and probably examine the RTL, as well. There is hope from the EDA world in this regard, though. Start-ups Zocalo Tech and NextOp Software recently introduced tools that could help with creating assertions. The Zocalo tool identifies signals for which, according to its heuristics, assertions would be appropriate. The NextOp tool goes a step further by synthesizing assertions. How well either tool would work with black-box third-party IP remains to be investigated.

In a less-desirable case, a port on the IP block does not follow any well-defined standard. In this case, as in creating your own assertions, the task comes down to the designer and the data sheets. “You must analyze the semantics of the interface as it is specified in the IP data sheet,” Wagner says. For complex ports this can be an error-prone and grindingly manual process.

Understanding how the ports on an IP block work is part of the challenge. Connecting them into the SOC design is a separate task that may require additional logic design—FIFO (first-in/first-out) buffers or protocol translators, for example. No synthesis tool can generate these interfaces. Ideally, as the IP block merges into the SOC, its test bench can contribute to the overall SOC-verification suite, generating stimuli and assertions for transactions that cross the boundaries of the block. But
this reuse of VIP is also a manual task begging for automation.

**The details devil**

Beyond the basic function of an IP block and the interfaces that connect it into the SOC, there are still more integration issues—for example, power management. SOC architects now have a bulging arsenal of techniques for reducing dynamic and static power at the chip level. These approaches include device-level techniques, such as a choice of transistors with different threshold voltages or with adjustable back biases to change their thresholds; circuit-level techniques, such as signal gating and fine-grained clock gating; and block-level tools, such as coarse-grained clock gating, voltage islands, DVFS (dynamic voltage-frequency scaling), and power gating. The issue for IP integration is how to coordinate the power-management strategy at the chip level with whatever assumptions the IP designers have made.

“It helps if the chip architects understand the power-management provisions in the IP blocks,” Madraswala says. “Can the block shut itself down? Does it need to be an independent power island? Is it capable of DVFS?” Designers should fold all this data-sheet information about the capabilities of the IP blocks into the power planning for the whole chip.

Once the project moves from planning to implementation, the influence may flow in the other direction. “For soft IP, the customer generally sets the strategy for power,” Wagner says. “Implementation work may fall to the IP vendor or the customer.” The point of using IP in the first place is to achieve a higher level of abstraction for the design team. So opening up the RTL to weave in a new power-management strategy is not a happy alternative. However, the chip-design team may sometimes be able to improve the power consumption of a block without understanding its structure.

Synthesis tools implement netlist-level power-saving techniques, such as signal or clock gating, requiring little work by the chip-design team. The team should always use logic-equivalence checkers, cautions Wagner, when modifying netlists with which they are unfamiliar. It’s easy for a tool to alter performance along with efficiency.

Approaches that alter the structure of the IP, such as power gating, may require either the active participation of the IP developer or reverse-engineering by the SOC team. Such changes may also require alterations to the test bench just to test the power-saving modes. Design-management-team members may have to balance how much energy they can save in a block against just how much abstraction they are willing to surrender to get the savings.

Scan insertion, another downstream automated task, should create no obstacles to black-box treatment of IP. Exceptions may exist, however. “High-performance-IP vendors tend to be very conservative about scan chains in their blocks,” Madraswala says. “They don’t want scan flip-flops on their critical paths.” Madraswala suggests checking the fault coverage you are getting rather than blindly trusting the vendor’s scan directives.

Madraswala also says that clock insertion should not require the chip designers to get too involved in the entrails of an IP block. All the necessary information to generate the clock structures should be in the deliverables, and the clock requirements should be in the data sheet. In this area at least, the box should remain black, as long as you don’t reorganize the clocks to accomplish coarse-grained clock gating.

Design closure Ideally, you could close the design—confirming timing, signal integrity, power
integrity, and design-rule compliance—without looking into the IP blocks. The IP vendor presumably has stated that the block will pass static timing analysis at the specified frequency, will be free of signal-integrity issues, and will be DRC (design-rule-checking)-clean. Because most IP today uses fully registered inputs and outputs, there should be no timing paths running across the boundaries of the block. So it should be no problem to hierarchically close timing.

The IP vendor bases its confidence on its own synthesis, placement, and routing using its own tools, scripts, and libraries, however. As the proverbial fine print says, your results may vary. “It may be necessary to resynthesize the IP to ensure it will meet timing using the customer’s standard-cell and SRAM libraries,” Wagner says.

The interesting question is what happens if something fails. If the design team has been successful in treating the IP block as a black box, they will be on unfamiliar ground in trying to resolve closure issues. The only way to preserve the integrity of the box will be to involve the IP developer in resolving closure issues. The developer, presumably, understands the netlist and the constraint files. If a novel use of the IP or changes to support new power-management modes also make the situation unfamiliar to the developer, then there may be no alternative to collaboration between the IP- and SOC-design teams.

Even within an organization, advanced power management can force close cooperation between IP and chip teams. “We use power gating in some situations,” says Harmel Sangha, senior director of marketing at LSI. “When we do, we depend on interpersonal relationships and regular reviews to convey our intent.”

“It really helps that these designs involve basically the same guys each time,” says Robert Madge, the company’s director of technical marketing. “If we are working with third-party IP, we will bring the supplier into our team. For example, we may put their IP on our own test chip and analyze the results with them.”

Surveying some critical steps in the IP-integration process, you can see that, in some cases, new tools are helping with automation. In other cases, there are opportunities for new tools to greatly ease the burden of the integration flow. The only way to preserve the abstraction of IP as a black box, however, may be to build an intimate, engineer-to-engineer collaboration between the IP-creating team and the SOC team. No tools are available to automate that relationship.

### For More Information

<table>
<thead>
<tr>
<th>Accellera</th>
<th>Open-Silicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppliedMicro</td>
<td>PMC-Sierra</td>
</tr>
<tr>
<td>LSI Corp</td>
<td>Qualcomm</td>
</tr>
<tr>
<td>NextOp Software</td>
<td>Xilinx</td>
</tr>
<tr>
<td>NXP</td>
<td>Zocalo Tech</td>
</tr>
</tbody>
</table>