FEATURE ARTICLE

Thomas Anderson

System-on-Chip Design with Virtual Components

Here in the Recycling Age, designing for reuse may sound like a great idea. But with increasing requirements and chip sizes, it's no easy task. Thomas explains how virtual components help suppliers get more mileage out of their SOC designs.

Design reuse for semiconductor projects has evolved from an interesting concept to a requirement. Today's huge system-on-a-chip (SOC) designs routinely require millions of transistors. Silicon geometry continues to shrink and ever-larger chips are possible.

But, the enormous capacity potential of silicon presents several challenges for designers. Design methodology and EDA tools are being severely stressed by SOC projects at the same time that narrowing time-to-market requirements demand more rapid and frequent introduction of new products.

SOC projects present another problem—how to design enough logic to fill up these devices. Few companies have the expertise to design all the intellectual property (IP) needed for a true SOC, and few have enough engineering resources to complete such a massive project. Even those with the required knowledge and plentiful resources may still be unable to finish a complete chip design in time to meet accelerated market demands.

The net result: SOC projects require design reuse. Only by leveraging off past designs can a huge chip be completed within a reasonable time. This solution usually entails reusing designs from previous generations of products and often leverages design work done by other groups in the same company.

Various forms of intercompany cross licensing and technology sharing can provide access to design technology that may be reused in new ways. Many large companies have established central organizations to promote design reuse and sharing, and to look for external IP sources.

One challenge faced by IP acquisition teams is that many designs aren't well suited for reuse. Designing with reuse in mind requires extra time and effort, and often more logic as well—requirements likely to be at odds with the time-to-market goals of a product design team.

Therefore, a merchant semiconductor IP industry has arisen to provide designs that were developed specifically for reuse in a wide range of applications. These designs are backed by documentation and support similar to that provided by a semiconductor supplier.

The terms "virtual component" and "core" commonly denote reusable semiconductor IP that is offered for license as a product. The latter term is promoted extensively by the Virtual Socket Interface (VSI) Alliance, a joint effort of several hundred companies to set standards for VC design, verification, and use. In this article, I describe the major virtual component (VC) types and discuss their use in SOC designs.

FORMS OF VC

VCs are commonly divided into three categories—hard, soft, and firm. A hard VC or hard macro is a design that is locked to a particular silicon technology. Such macros are fully placed and routed and are available in a fixed size and format.

They can be easily dropped into the floorplan for a chip in the same target technology, because the silicon technology is known, and they usually have predictable timing. However, they can't easily be mapped to another silicon vendor (e.g., a second source) or even to a different technology from the same vendor. The VC user also

has little or no choice in terms of feature set modification or customization.

Hard macros are most often provided by ASIC and FPGA vendors as part of their library. Such macros are a natural extension to the basic cell library used to implement the VC user's design. Because the silicon vendor sells chips, there's no incentive to provide a VC in a more portable form that makes it easier for the customer to switch to another supplier.

Some IP vendors, especially those supplying microprocessor and DSP designs, also provide a VC in hard form. This option shows that the key elements of processors, especially data paths for arithmetic computation, are often designed at the transistor level for maximum performance.

Some processors, as well as many other kinds of VC products, are available from IP vendors in soft (or synthesizable) form. A VC described in Verilog RTL or VHDL code gives the user maximum flexibility. It can be mapped to virtually any target ASIC or FPGA technology using commercial logic synthesis tools.

The user may also be able to control the VC feature set, for example, by setting variables in the code or by running a utility that modifies the code under user control. Of course, because the user licenses the actual Verilog or VHDL source code, it can always be modified directly.

One issue with a synthesizable VC is that the precise timing is not known until the VC is mapped to a target technology. Accordingly, soft VC suppliers must synthesize to a range of representative target libraries and ensure that timing is satisfied.

The supplier may also have to supply guidelines to assist the user in laying out the chip containing the VC so that the postroute timing is still correct. Such guidelines may include recommendations for target technology, pin assignments for external I/O, floorplanning for key modules, and routing of critical paths.

The definition of a firm VC varies widely. The term is used most commonly to refer to a soft VC accompanied by an example layout-level implementation, although some people refer to a VC as firm whenever it comes with layout guidelines. The term also refers to a netlist-level VC that has been mapped by synthesis to a target technology but is not yet placed and routed.

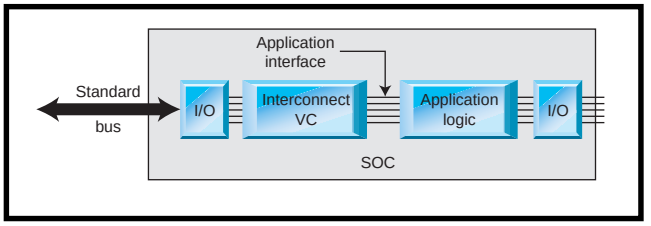

Figure 1—*A soft VC interconnect fits between the application logic and the I/O signals. Implementation instructions generally include guidelines for connecting the interconnect to the external chip pins.*

It is possible, although difficult, to make customizations to a VC in netlist form. Synthesis tools can also provide some degree of portability to new technologies, but the range of optimizations available when synthesizing from the netlist level is more limited than from Verilog or VHDL.

VC FUNCTIONS

Numerous factors can lead to a decision to license a commercial VC for inclusion in an SOC design. The expertise and resources available and the time-to-market requirements for the end product must be balanced against the expense of the VC license. Even a company with vast, expert resources may be able to produce a better product faster by leveraging external IP.

This is especially true if the VC implements a common function because little is gained by designing and optimizing such a function rather than focusing on product-differentiating features. For example, the VC may duplicate the function of existing stand-alone chips (e.g., a UART or a floppy disk controller) or implement a common arithmetic function such as a multiplier.

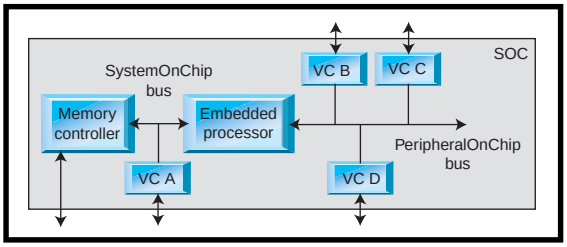

Figure 2—*System-on-chip designs may contain both a system bus connect and a peripheral bus connect. Custom I/O blocks that provide functions not commercially available may also be included.*

Perhaps the highest leverage is provided by a VC that implements a formal or de facto standard. Because many types of chips and end products must meet a standard, a VC that implements this standard is ideal as a commercial IP product. It's rare that an end user can add enough value with an in-house design to offset the time savings and standards expertise embodied in a well-designed VC.

The standards provided by VCs range from formal IEEE, ANSI, and IEC specifications to new technologies. Examples include communications protocols like Ethernet and ATM, computational functions such as MPEG and JPEG, parallel interconnect standards such as PCI and AGP, and serial interconnects like USB and IEEE 1394. These examples have wide applicability to many different types of SOC-based products, and the standards themselves are well enough defined to allow implementation as a VC.

A VC implementing an interconnect technology probably has the widest range of application. For example, PCI is used in diverse types of electronic products. Although it was developed as a personal computer peripheral bus, PCI has now been adopted for workstations, mainframe computers, military applications, and networking/telecommunications systems. USB is following a similar expansion of scope beyond the PC, as it is used to connect peripherals to gaming systems, set-top boxes, and PDAs.

The widest penetration of all may occur with 1394, which is designed to interconnect both computers and diverse consumer electronics devices. Products available today with 1394 support include video cameras, digital televisions, digital VCRs and professional audio equipment. Although it is not yet supported in mainstream PC chipsets,

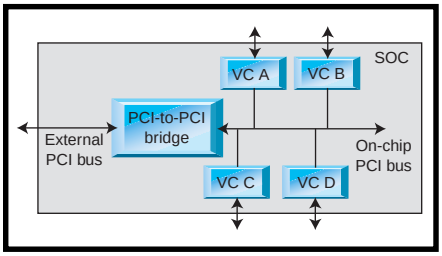

Figure 3—*A multiprotocol I/O controller can be enabled using PCI as an on-chip bus. In this application, a PCI-to-PCI bridge supports multiple I/O technologies using a single PCI load or slot.*

many desktop and laptop computers offer 1394 support and such peripherals as disk drives and videoconferencing cameras are starting to appear.

USING A VIRTUAL COMPONENT

The path for a chip designer to use a VC depends on both form and function. A hard macro can be dropped into a chip layout fairly easily, as long as the macro and chip use the same silicon technology. But, simulation and timing analysis with a hard macro is not as simple.

Generally, the VC supplier must provide separate simulation and timing models. Correlation of these models to the hard-macro implementation may be a difficult problem for the VC provider and a potential issue for the user. A VC at the netlist level has fewer problems, although the inefficiency of gate-level simulation may require the VC supplier to provide a high-level model in addition to the netlist.

A soft VC has several advantages in terms of design flow because it can usually follow the same design process as the rest of the chip. The user runs synthesis to map the RTL design to the target technology, uses static timing analysis to verify timing, lays out the chip following the supplier's guidelines, and reruns static timing analysis with back-annotated postroute delays.

The VC also has the same advantages as any RTL design in that the source code also serves as the simulation and timing model. The lack of perturbation to the user's design methodology is a key attraction for a synthesizable VC.

A VC with no requirements for connection to chip pins, such as a fully embedded processor, is wired into the chip design like any other module. An interface VC, however, generally has some I/O signals that need to connect to external chip pins. The implementation and layout instructions for a soft interconnect VC generally include guidelines on how to connect to the pins.

As shown in Figure 1, such a VC essentially fits in between the chip pins and the user's application logic. The set of VC I/O signals to which the user connects is often referred to as the application interface.

MULTIPLE VC APPLICATIONS

It's becoming common for an SOC design to use more than one VC. Although there may be no direct interaction between one VC and another, in other cases they may be linked on a common bus. The term "on-chip bus" (OCB) describes a formally specified bus that interconnects multiple VC blocks within a single chip.

An OCB is likely to fall into one of two categories—system or peripheral bus. A system bus connects an embedded processor or DSP with the memory controller and higher speed I/O devices. A peripheral bus connects to lower speed I/O technologies.

An interface block generally bridges these two buses, although some embedded processors directly drive both buses. In SOC designs with multiple embedded processors, the processors generally communicate over the system bus.

Figure 2 shows an SOC that has both system and peripheral OCBs. In an actual chip, the system bus might link to a 400-Mbps 1394 interconnect VC and the peripheral bus would support slower I/O technologies such as USB, RS-232, and IrDA (infrared). It's also possible for the SOC designer to create custom I/O blocks that connect to an OCB to support functions not available from commercial VC sources.

It would be nice if widely adopted OCB standards existed, but this is not the case. Nearly every embedded processor has its own proprietary system bus; some have defined proprietary peripheral buses as well. This situation can make it difficult to take a VC designed for an SOC with one embedded processor and move it to a different chip. A few buses (e.g., AMBA buses for ARM processors) are supported widely enough to be considered a de facto standard in some application spaces.

One interesting option for a peripheral OCB is an on-chip version of PCI. Several popular embedded processors are available in versions that provide PCI support, and many interconnect VC families include an option for PCI support on the application interface. Using a PCI OCB also enables existing PCI-based chips to be easily transformed into macros for use in larger SOC designs in newer technologies.

The size of a PCI VC, which usually ranges from 7k to 15k gates, is not a major issue in the context of a million-gate SOC. Other objections to PCI as an OCB (e.g., its use of tristates and multiplexed address/data lines) can be addressed by using a PCI derivative.

In fact, a number of SOC designers use PCI or a derivative bus as an OCB. Figure 3 shows one interesting application, a multiprotocol I/O controller.

This design allows multiple I/O technologies (e.g., USB, 1394, and Ethernet) to be combined by using a VC with PCI for each and then using a PCI OCB to link the VCs together. A PCI-to-PCI bridge permits this wide range of I/O support while using only a single PCI load on the motherboard or a single PCI slot in the system.

VSI is tackling the OCB issue by defining the virtual component interface (VCI), a standard application interface for VC designs done in-house or available from commercial IP suppliers. VCI is not an OCB but a standard VC interface that enables OCB usage.

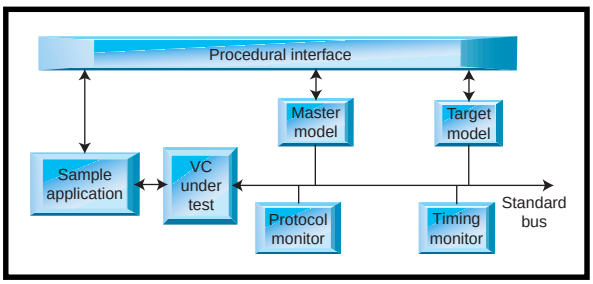

Figure 4—*A verification environment provides behavioral models and test scripts to verify the functionality of the VCs. This approach can be used in full SOC verification as well as stand-alone VC verification.*

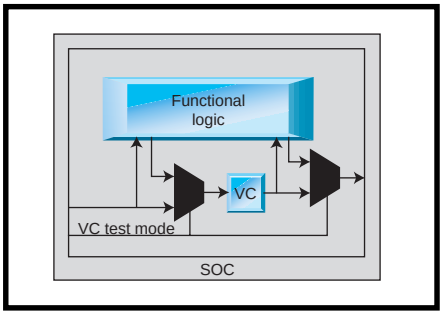

Figure 5—*One test method for legacy VCs is a parallel access test process. This process uses multiplexers to bring all inputs and outputs to the external pins and plays a predefined set of test vectors.*

The idea is that the SOC designer needs to develop only a bus translator from VCI to the chosen OCB. With this translator, any VCI-based block can easily be connected to a given OCB. VCI has been defined for easy translation to popular OCBs (including PCI) and translators for such buses will be available as licensable IP.

VC VERIFICATION ISSUES

One area common to both single-VC and multiple-VC SOC designs is the need to verify and test the complete chip. The SOC design and test engineers want to leverage and build on the verification and test done for each individual VC, and accordingly they expect the VC provider to assist in this process.

As previously noted, each VC is accompanied by some sort of simulation model that enables the SOC designer to run chip-level tests that involve the VC. But, that doesn't provide any support for determining whether the VC is connected properly in the design and is operating correctly in the context of the full SOC. A VC with a miswired application interface will simulate, but the results may not be correct.

Many suppliers address this problem by providing a verification environment along with the VC itself. Such an environment provides behavioral models (usually in Verilog, VHDL, or C) that interact with the VC and a set of test scripts that use these models to verify the functionality of the VC. Figure 4 shows a sample verification environment for an interconnect VC such as PCI or 1394. Key components of this approach include:

- master model to address the VC as a target
- target model to respond to VC as a master
- sample application code to stimulate VC
- tests written as scripts of procedural calls
- monitors for protocol and timing correctness

As a stand-alone test for the VC, a verification environment lets you verify a postsynthesis netlist or a customized version of the VC to ensure that protocol and timing rules are satisfied. A simple set of test vectors performs much the same function for stand-alone VC verification, but debug is harder without monitors and readable test scripts.

One of the main advantages of the verification environment approach is that many of its components can be used in the full SOC verification in addition to the stand-alone VC verification. The master and target models can be connected to the SOC bus interface while the protocol and timing monitors can continue to be used.

It may be possible to continue to run the same test scripts on the SOC, requiring the chip designer to modify the procedural interface to stimulate the VC from the actual SOC logic rather than from the sample application.

A verification environment can be provided with any VC, whether in hard or soft form. Synthesizable-VC suppliers usually provide verification environments and hard-macro suppliers often provide some components such as bus models to aid in SOC verification.

Many of these components are useful to a designer developing a custom implementation of an interface or processor. So, it's quite common for IP providers to license a verification environment even to customers who do not license the VC itself.

VC TEST CHALLENGES

By its nature, a VC is embedded into the SOC design by the VC user. Once a VC is inside the larger chip design, it's no longer accessible as a stand-alone functional block. Whatever test vectors or methods the VC supplier provides can no longer be used without

special considerations to design-for-test (DFT) approaches during chip design.

The challenges of embedded VC tests depend on the nature of the VC itself. A synthesizable VC is the easiest case. The VC user runs synthesis to map the Verilog or VHDL code to the target technology, lays out the chip following the supplier's guidelines, and runs timing analysis with back-annotated postroute delays.

The result, as I noted, is that the VC follows the same design process as the rest of the chip. The same is generally true for chip test methodology, since virtually all test insertion tools run on the postsynthesis netlist. Whatever approach the VC user takes for the rest of the chip—full scan, partial scan, built-in self test (BIST), or JTAG—is usually applied to the soft VC also.

To ensure that the user has no problems, a soft-VC supplier should use a clean design style with DFT in mind. Typically, soft VC designers use simple clocking schemes, avoiding latches and gated clocks unless they are required.

For example, the USB protocol has suspend and resume commands that put a peripheral device into a minimal-power state. Some amount of clock gating is unavoidable in a USB device VC because of this requirement.

In contrast, a hard macro user is stuck with whatever test technique (if any) is built into the VC. If the VC includes full scan, partial scan, or BIST technology, it's helpful if the remainder of the chip also uses this approach. Integrating a VC scan chain into the full-chip scan chain is usually a simple matter of running scan insertion and stitching tools.

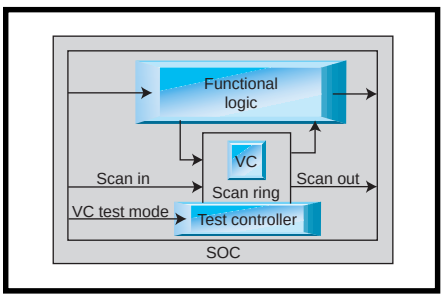

Figure 6—*Another approach for testing legacy VC involves serial access to the virtual-component I/O signals. The legacy VC is surrounded with a JTAG-like register chain that can drive VC inputs and read VC outputs. The disadvantage of this method is that test times can be very long for complex blocks.*

Sometimes the hard macro includes no internal DFT at all (often called legacy VCs because the user needs to treat them as black boxes in terms of testing). Generally, the VC supplier provides a set of test vectors, perhaps guaranteed to provide a certain level of coverage as defined by the single stuck-at fault (SSF) model. Of course, running this exact set of tests on the VC once it's embedded in a chip can be a challenge for the VC user.

When the only test method available for a legacy VC is "playing" a set of predefined test vectors, two approaches are common. The first is simply to bring all VC inputs and outputs out to external chip I/O pins using multiplexers as shown in Figure 5.

The parallel test-access process can be automated by test-insertion tools and requires only a VC test-mode pin setup prior to running functional vectors on the VC. This approach is attractive for interconnect VCs like PCI because some VC inputs and outputs will already be connected to chip I/O pins for functional reasons.

This method breaks down if there are more VC inputs and outputs than chip pins available. Staging registers may be needed to accrue each complete VC vector over several clock cycles. If the test vectors are also intended to check VC timing, inserting multiplexers into the path adds delays and may require changes to the timing vectors.

This approach doesn't work at all for analog VCs unless some sort of analog multiplexer is available. Intervening digital logic makes it impossible to apply or measure continuous analog values.

The second approach, called internal boundary scan or VC isolation, surrounds the legacy VC with a JTAG-like register chain that can drive the VC inputs and read the VC outputs. As shown in Figure 6, this setup requires some form of TAP-like test controller to run the scan chain.

Because this technique relies on serial access to the VC I/O signals, test times can be long for complex blocks like embedded microprocessors. So, IP suppliers are developing methods to test complex VCs using existing functional datapaths, including on-chip buses.

CORES IN FPGA DEVICES

As I mentioned, it's common for FPGA and ASIC vendors to provide hard macros for common core functions. Traditionally, these offerings have been limited, but the increasing speed and size of FPGAs means a broader scope of core offerings.

For example, 33-MHz PCI cores are readily available, and some FPGA vendors even claim fully-compliant 66-MHz cores. Such cores require a great deal of hand-tuning during the design and layout stages so they are optimized for a particular technology.

Like their ASIC counterparts, FPGA designers may desire flexibility and portability and can therefore benefit from synthesizable designs. There's no reason that synthesizable cores can't be targeted to FPGA designs, but mapping a core to an FPGA technology does present challenges.

In general, commercial synthesis tools are less efficient at mapping to complex programmable logic blocks than to relatively simple ASIC cell libraries. The effect of routing length is usually greater for FPGAs, producing unanticipated delays on critical paths.

The design-tool flows for most FPGA vendors do not have a tight loop from layout back to synthesis. So, the synthesis process usually can't take into account useful layout information (e.g., a chip floorplan specifying the location of timing-critical blocks.

The result: less correlation between the preroute timing estimates from the synthesis tool and the accurate post-route timing results. When it comes to final chip timing, surprises are usually negative rather than positive.

Finally, the gate capacity of even the largest FPGA devices is far below that of ASICs and custom chips, which limits opportunities for multicore designs. So, on-chip buses aren't common in FPGAs.

Combinations of a few cores are possible in large programmable devices: for example, including several Ethernet cores to implement a network repeater, or pairing a PCI core and a USB host core for an adapter chip to add USB to a PC without chipset support.

WORKS IN PROGRESS

Design reuse, including licensing of commercial IP, is key to SOC design. And a significant industry has arisen to provide a wide range of VC products. Several industry initiatives are addressing the needs of VC suppliers and users.

The IEEE Test Technology Technical Committee Embedded Core Test Study Group has also been working on VC test issues, and this has led to the proposed IEEE P1500 specification.

It's important not to minimize the issues and concerns involved in VC use. Recognizing this, two industry groups focus on the business and legal aspects of VC license and use.

Many VC suppliers are members of the RAPID trade association, which works on common VC license agreements and catalog methods. RAPID cooperates with the Virtual Component Exchange (VCX), which is developing a structure for simplified VC transactions.

The immaturity of EDA tools for VC integration and SOC design is one challenge to design reuse. Also, new suppliers may underestimate the difficulty

of designing for reuse across a diverse customer base. Some VC types (e.g., interconnect cores, embedded processors) need a vertically integrated supplier that supports software and hardware.

Despite these issues, virtually every major system and semiconductor manufacturer has licensed external IP and employs internal reuse. Commercial VC products are incorporated into thousands of chip designs, and the many successful products on the market prove the value of this approach.

Thomas Anderson is director of engineering in the Semiconductor IP Group at Phoenix Technologies. He is also a member of the PCI special interest group steering committee and chairperson of the 1394 Developers' Conference. You may reach him at tom_anderson@phoenix.com.

REFERENCES

T.L. Anderson, "Interconnect Solutions for Embedded Systems," *Microsoft Embedded Review*, 1 Mar. 1999.

T.L. Anderson, "The Reality of Using Cores as Virtual Components," *Electronic Engineering*, July/Aug. 1998.

T.L. Anderson, "Make vs. Buy: A Case for Licensing Cores," *Silicon Strategies* **1**, Feb. 1998.

T.L. Anderson and C. Stahr, "ASIC Design Flow with Synthesizable Cores," Proc. of Design, Automation and Test in Europe: User's Track, Feb. 1998.

T.L. Anderson, "Thoughts on Core Integration and Test," Int'l Test Conference 1997 Proc., Nov. 1997.

T.L. Anderson, "CPLD Cores can Speed Design Turnaround," *EE Times*, 21 Apr. 1997.

T.L. Anderson, "The Challenge of Verifying a Synthesizable Core," *Computer Design*, July 1996.

T.L. Anderson, "RTL Cores Promote Design Flexibility," *EE Times*, 15 May 1996.

S. Brandt, "Building a Multimedia Video Conferencing Chip with a Synthesizable PCI Core," *Integrated System Design*, Sept. 1997.

S. Davidmann, "Using Pre-designed Components in PCI Bus-based Systems," *Electronic Engineering*, May 1996.

IEEE, "A D&T Roundtable: Testing Embedded Cores," *IEEE Design and Test of Computers*, May/June 1997.

IEEE P1500 Standards for Embedded Core Test, grouper.ieee.org/groups/1500.

A. Oldham, M. Knecht, and T.L. Anderson, "IP Cores in AGP Bus-based Systems," *Electronic Product Design*, May 1998.

SOURCES

VSI Alliance
(408) 256-8800
Fax: (408) 356-9018
www.vsi.org

RAPID
(408) 341-8966
www.rapid.org

Virtual Component Exchange
+44 1506 404100
Fax: +44 1506 404104
www.vcx.org