Core Design and System-on-a-Chip Integration

ANN MARIE RINCON
CORY CHERICHETTI
JAMES A. MONZEL
DAVID R. STAUFFER
MICHAEL T. TRICK
IBM Microelectronics Corp.

IBM's experience with core-based designs provides insight into methodology, SOC design styles, core design trade-offs, and ASIC design processes. The authors describe a prototype cosimulation system developed for the PowerPC core and present SOC designs to illustrate their methods.

ALTHOUGH SYSTEM-ON-A-CHIP (SOC) design takes various forms, we can identify three main approaches (as described by Susan Mason of Information Architects, speaking at the 1996 Design Automation Conference): vendor design, partial integration, and desktop. These approaches provide a wide range of design flexibility and time-to-market scenarios. In the vendor design approach, an ASIC vendor or a design services group designs the core and ASIC to meet a customer's functional specification. This approach gives the system designer little or no control over the design schedule and may result in a longer time to market and less design flexibility than other approaches. Although vendor design can lead to the lowest production cost per die, it may also result in high engineering costs, thus lending itself only to high-volume applications.

Using the partial integration approach, the system designer creates most or all the ASIC gates, and the ASIC vendor designs and integrates the core with the customer's ASIC logic. This approach provides a more flexible division of labor, and the system designer has some control over the design schedule.

But the desktop approach gives the system designer the most flexibility and enables the fastest time to market. In this scenario, the ASIC vendor designs the core, and the designer builds the ASIC logic and integrates the core, using a standard ASIC design flow. The ASIC vendor is no longer the design schedule gatekeeper. The desktop approach incurs the lowest engineering cost and is suitable for lower-volume solutions.

In the past, vendor design and partial integration were the dominant approaches. Recent strides in core modeling and ASIC design methodology allow system developers to choose any of the three approaches. They must consider many variables in selecting an approach appropriate for a project and/or a design team. For example, designers capable of taking control of the process should have that option—that is, for them, the desktop approach is most appropriate. From the core developer's point of view, designing a core to support the desktop approach facilitates all three scenarios.

Here, we describe IBM designs that cover the spectrum of SOC design approaches. Two designs used the desktop approach. The third design, developed by both the ASIC vendor (IBM) and the customer, represents the partial integration approach. The fourth design, implemented entirely by IBM to meet the system designer's functional specification, exemplifies the vendor design approach.

Hard cores (see "Terminology" box) in these designs include a 32-bit PCI (peripheral component interface), a PowerPC 401 CPU, a memory controller from Rambus, Inc., a video PLL (phase-locked loop), a video DAC (digital-analog converter), and two high-speed SRAMs. Several soft cores serve either as standalone functions or to integrate several hard cores into a single function. All the cores are 0.35-µm technology implementations.

Design-for-integration techniques

Core design techniques such as parameterization, function partitioning, on-chip buses, built-in modularity, and use of firm cores address the challenges of integrating cores into ASICs.

0740-7475/97/$10.00 © 1997 IEEE

INTEL-1049
U.S. Patent 7,151,430

Parameterization. Parameterization is a key technique for customizing soft cores. By using VHDL generics or Verilog parameters and the gate-level netlist, designers can select or eliminate commonly customized features during synthesis. A parameter set defined and verified by the core developer limits allowed modifications, greatly reducing the need for subsequent reverification by the customer and simplifying or minimizing the tasks of supporting multiple core configurations. Examples of features that can be customized through parameterization are the core base address, baud rate clock, bus interface selection, number of DMA channels, and number of FIFOs.

Function partitioning. An alternative to implementing a function as a single, monolithic hard core is to partition it into multiple hard, firm, and soft cores. Portions of the function that are timing-critical (such as a CPU) or function-critical (analog elements) are best implemented as hard cores. A firm-core implementation is appropriate for a function without timing requirements that dictate custom layout, but requiring intellectual property encryption. Functions that do not require fixed layout or IP encryption, or that are candidates for frequent customization, are best implemented as soft cores.

The reduced size and increased flexibility of the core elements resulting from partitioning alleviate chip-level routing problems that occur with large, monolithic hard cores. The core developer can provide the partitioned functions to the designer as individual cores or as a top-level soft-core netlist containing the hard-, firm-, and soft-core submodules. Designers can modify the top-level netlist by following guidelines set forth in each core's documentation.

An example of effective function partitioning is the RAMDAC core, which originated from the IBM 526DB palette DAC standard product chip. The palette DAC function consists of digital logic, high-speed SRAMs, several analog components, two PLLs, and a 10-bit video DAC.

It is possible to implement the palette DAC as a single, monolithic hard core. But doing so complicates floor planning, forcing the layout engineer to work with a large, inflexible block with little wiring porosity. The core must be close to the appropriate chip signal and test pads. It must not block signal wires in other logic blocks on the chip. And the sensitive analog circuitry must not have any interference from noisy signals routed near the core. Partitioning the palette DAC makes it easier to place each subcomponent next to the required pad locations without blocking other signals or receiving interference.

The single-core implementation prevents chip designers from customizing the function by substituting custom logic for the IBM digital portion of the palette DAC. We decided to partition the standard product function into four distinct

Terminology: core types

An industry standard for core terminology is just beginning to emerge. It may differ from the terminologies in use at IBM and other companies, and likewise those terminologies may differ from each other. Therefore, we define the following terms as we use them in this article.

A *synthesizable core* comes in a technology-independent high-level description language form. The core's layout is completely flexible, but its speed and density are limited by the characteristics of the ASIC cell library in which it is implemented. Synthesizable cores require ground-up synthesis, test, static timing analysis, and user verification. Their flexibility limits their "drop-in" design usability and their ability to leverage performance and area optimization characteristics.

A *soft core* is a technology-dependent gate-level netlist. In some cases, a soft core also includes a small amount of high-level technology-independent code that a designer can use to parameterize the core during synthesis. Because of the core's technology-dependent nature, its size and speed are more predictable than those of synthesizable cores. The soft core's layout is flexible, but floor-planning guidelines may be necessary to achieve performance targets.

A *firm core* reaches the customer in the form of an encrypted or abstracted black box that protects the core's intellectual property content. Designers incorporate a firm core into an ASIC design in the same manner as a library element. The ASIC vendor lays out the core, using a technology-dependent gate-level netlist. This provides flexibility in the chip layout process because the core form factor is not fixed. A firm core's size, aspect ratio, and pin location can be changed to meet a customer's chip layout needs, and floor-planning guidelines assist the chip designer in making trade-offs. The technology-specific nature of a firm core makes it highly predictable in performance and area. And, because layout uses a gate-level netlist, a firm core has the same porosity and routability as a soft core.

A *hard core* is also an encrypted or abstracted black box, which designers incorporate into an ASIC design in the same manner as a standard cell library element. Unlike firm cores, however, hard cores have a fixed, custom physical layout. The technology-specific layout allows maximum optimization in terms of performance and density. Hard cores, however, have the most limited vendor portability and greatest difficulty of reuse when moving to a new process technology. A hard core may contain significant routing blockages (poor porosity), making the placement of other blocks and chip-level routing difficult.

INTEL-1049
U.S. Patent 7,151,430

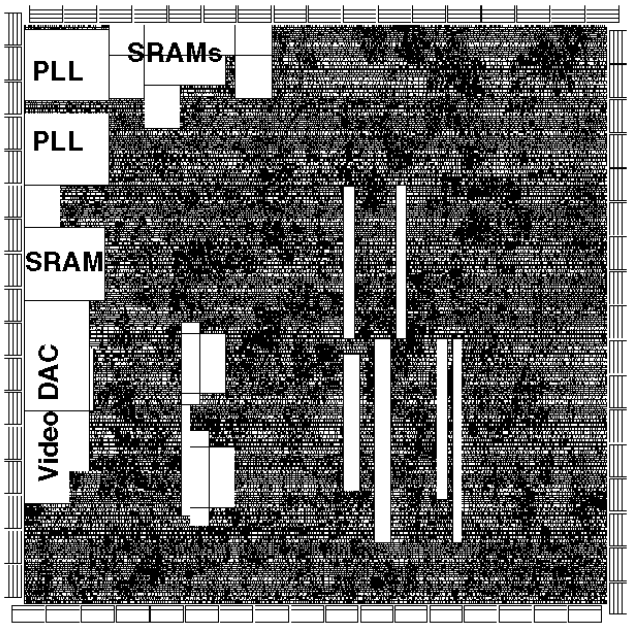

Figure 1. Partitioned RAMDAC core integrated with customer logic. Unlabeled blocks are SRAMs and register arrays.

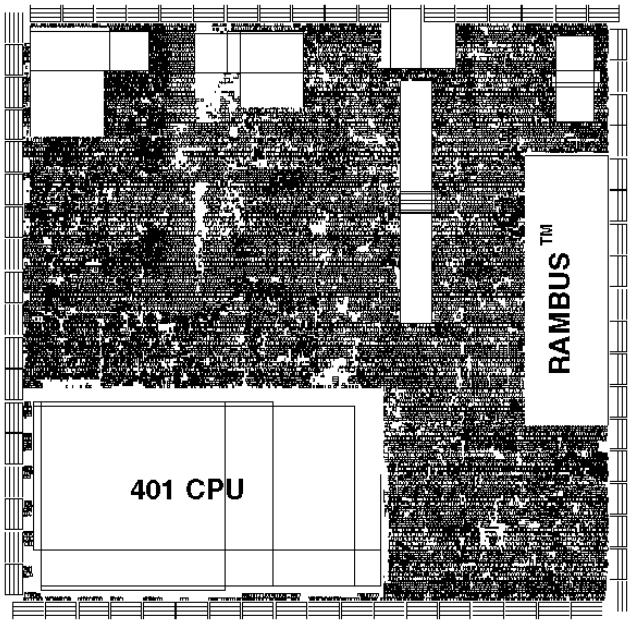

Figure 2. PowerPC-core-based design. Unlabeled blocks are a PLL, RAMs, and register arrays.

blocks that designers could use either separately or as an integrated solution. We implemented the video DAC, PLL, and high-speed SRAM as individual hard cores in the range of 10,000 to 30,000 cells each. We implemented the remaining digital logic as a soft-core netlist (RAMDAC) that included the other hard cores as components and was the delivery vehicle for the integrated solution.

Figure 1 shows an example of the successful use of the RAMDAC integrated solution. Other chip designs, requiring only the PLL, video DAC, or SRAM, could use these cores as stand-alone functions without sacrificing valuable silicon to unused palette DAC functions.

Another example of multiple cores derived from a standard product chip is the PowerPC core product line (based on the PowerPC 40X chip series). We divided the PowerPC microcontroller chip into a hard core and several soft cores. The timing-critical CPU became a hard core; peripheral functions such as the DMA controller, external bus interface unit (EBIU), timers, and serial port unit (SPU) became soft cores.

The first chip to use these PowerPC cores contained the 401 CPU hard core and the SPU soft core (Figure 2). It did not use the off-chip memory interface core (EBIU). The application called for the Rambus high-speed memory interface, provided as a separate, mixed analog-digital hard core. Because we had partitioned the 40X PowerPC function into multiple individual cores, the customer could select and use only the functions required for the design.

Interfacing with on-chip buses. Designers can use standard on-chip buses to eliminate the suboptimal glue logic required to integrate one or more cores with customer logic or with each other. Successfully used in the world of printed circuit board design for several years, this method is extendable to SOC designs. Standard buses ease integration of peripherals and independent design of user modules by providing a standard interface and communication protocol. Core logic and customer logic designed to a consistent protocol can quickly interconnect without requiring additional glue logic gates.

As in board design, the latency, bandwidth, and interface compatibility trade-offs among blocks of differing traffic characteristics dictate the need for a hierarchy of buses. For the PowerPC cores, IBM devised a dual on-chip bus architecture (Figure 3): The processor local bus (PLB) serves high-speed devices, and the on-chip peripheral bus (OPB) serves lower-speed peripheral devices. A separate core provides arbiter logic for each bus, and a bridge core transfers data between the two buses to further enhance usability.

In addition to the PowerPC peripheral cores, many other cores interface to the PLB/OPB bus structures. These include a UART (universal asynchronous receive transmit), a time division multiplexer, a universal serial bus, an Ethernet, an HDLC (high-level data line controller), a MAL (memory access layer), an IIC (interintegrated circuit) serial bus interface, and an IEEE 1284 parallel port unit.

Figure 4 represents the architecture of a design using the

INTEL-1049
U.S. Patent 7,151,430

401 core, several PowerPC soft-core peripherals, and cores specific to wired communication applications. All cores interface to each other through the PLB/OPB bus structures. The PowerPC EBIU and code decompression cores connect directly to the PLB bus; the Ethernet, HDLC, and MAL cores interface with the OPB. The chip designer chose a custom UART design and created a general-purpose I/O macro, designing these blocks for the OPB interface and verifying them with the PLB/OPB model toolkit.

Model toolkits for standard board-level buses such as ISA, EISA, PCI, and SCSI have been available for several years to help chip designers verify their bus communication logic. These toolkits are equally important to the SOC designer, providing an efficient design environment by allowing design and verification of macro functions without simulation of the complete project. The PLB/OPB toolkit includes bus master and slave models and a bus arbiter model for each bus. Also included is a bus monitor for verifying all bus transactions. Using this toolkit, the designer of the chip in Figure 4 saved an estimated two to three months of design and verification time.

Predesigned cores, which constitute over 90% of the chip logic, shortened the design time by an estimated 50% to 75%. Wired in four levels of metal, the design achieved over 80% utilization of available cells on the die. Only the CPU core presented significant blockage to the layout system. All the other cores, which were soft, were flat-routed with standard layout software and techniques.

Hard-core modularity. Core developers can provide design flexibility, even for a timing-critical hard core, by partitioning its function into several small, individual hard cores, or "chiplets." The timing-critical layout blocks are retained in smaller subblocks, enabling the designer to customize the core by selecting only the needed chiplets. Automatic generation of placement records, based on the selected configuration, maintains timing at the core's top level.

Meeting timing constraints on the 32-bit, 66-MHz peripheral component interface core shown in Figure 5 (next page) required a fixed layout, which dictates a hard-core implementation. Because the PCI function is highly configurable, we partitioned the design into nine chiplets that can be individually selected as required by an application.

IBM customers have designed several chips containing chiplets, using the desktop approach. For example, one design requiring a PCI controller used a PCI engine chiplet in combination with register configuration, master FIFO write server, and master FIFO read server chiplets. Another design requiring a PCI bridge also used the engine and register configuration chiplets but no FIFO servers. Instead, it used the master DMA write server and master DMA read server chiplets. The PCI function implemented in the design using

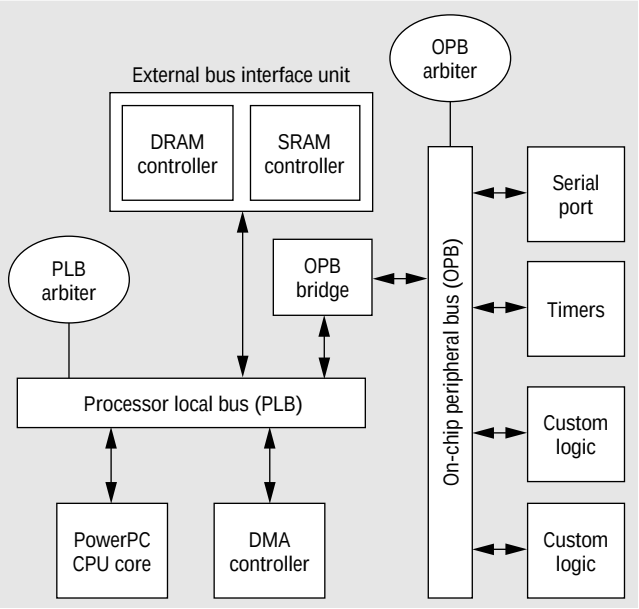

Figure 3. *Dual on-chip bus architecture.*

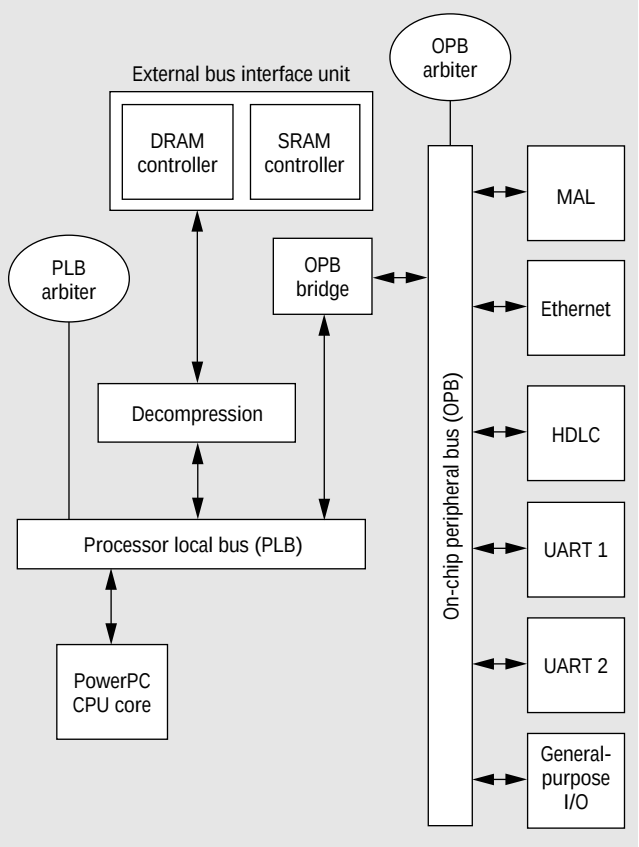

Figure 4. *On-chip bus implementation.*

INTEL-1049
U.S. Patent 7,151,430

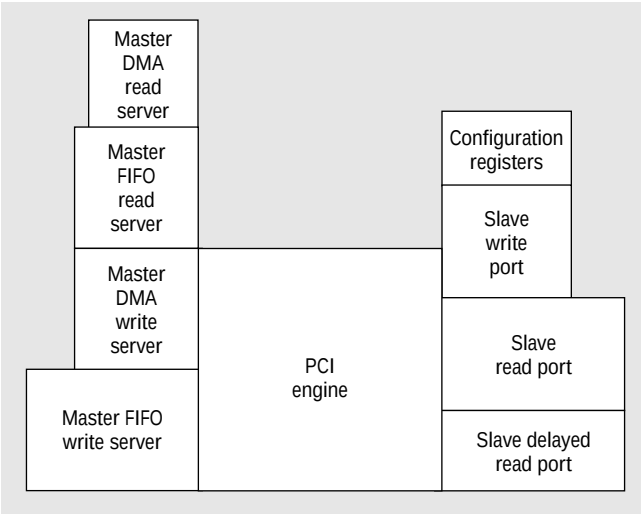

Figure 5. *PCI chiplets.*

the FIFO servers was approximately 35% smaller than a single hard core containing all nine chiplets. The PCI function using the DMA servers was over 40% smaller.

Firm cores. Using firm cores increases design flexibility and reduces layout problems. A core provider who requires IP protection for functions without critical timing requirements that drive a fixed hard-core layout should implement such functions as firm cores. Firm cores provide abstracted core views to the designer; the vendor replaces the firm cores with the gate-level netlist during chip layout. By allowing core size, aspect ratio, and pin locations to be altered during layout, this method facilitates an optimal chip layout. It alleviates tiling problems and reduces the unused silicon space that sometimes occurs when several large cores reside on the same chip.

SOC design methodology

Figure 6 illustrates an ASIC design methodology proven on very large (500,000 to 2 million gates), high-performance (over 100 MHz) designs. IBM ASIC customers used this methodology to create the designs described in this article. The most significant factor in achieving success on these large, complex designs is the sign-off criteria. We base design sign-off on static timing analysis and DFT compliance with fully automatic test pattern generation rather than exhaustive delay simulation and functional manufacturing test vectors. We enforce DFT rules through test structure verification (TSV) software provided by the ASIC vendor.

We run the TSV against the design at both the release-to-layout and release-to-manufacturing checkpoints. Similarly, we use a "golden" static timing analysis tool (Einstimer) and library for timing sign-off at these same checkpoints. We perform additional technology-specific manufacturability checks using an IBM-developed set of checking routines.

Functional verification of the design is left entirely to the designer. IBM does not resimulate ASIC designs before or after layout. We provide an updated postlayout design netlist to the ASIC customer for functional verification, with an SDF (standard delay file) on request, to support delay simulation. While many customers still rely on gate-level simulation for functional verification after layout, we recommend the use of formal verification tools for this task.

We have functionally verified pre- to postlayout versions of designs exceeding 800,000 gates with a 24-hour turnaround using the DesignVerifyer tool from Chrysalis. With the advent of chips containing complex embedded cores, verification through gate-level simulation has become even less practical, and formal verification methods have become more essential.

Our timing-driven layout system handles both hierarchical and flat designs. Timing assertions (description of expected arrival times, clock cycles, false paths, and so forth) for static sign-off generate timing targets for the place-and-route system. To close the final postlayout chip timing, we use a series of in-place optimization tools for drive strength optimization, buffer insertion, clock tree placement, and scan chain reordering.

Figure 6 also shows how we have updated the basic ASIC methodology with core-specific deliverables. Although the basic flow and sign-off points remain the same, the design kit has changed substantially. In addition to the base library of ANDs, NANDs, latches, and SRAMs, the ASIC vendor now supplies large pieces of the customer design in the form of soft-core netlists and black box models for firm- and hard-core functions. This requires the creation, delivery, and support of several new core-specific models. New tools and design techniques are also needed to address the verification requirements of complex cores (such as embedded controllers) that now reside on ASIC silicon.

Simulation

The customer receives soft-core functions as gate-level netlists. These netlists are mapped to the same ASIC library used by the customer logic. Simulation of the core netlist, therefore, requires no unique support beyond the design kit provided for customer logic simulation at the gate level.

Simulation models for hard and firm cores fall into two major categories: full-function models (FFMs) and bus function models (BFMs). Each hard- or firm-core macro requires an FFM. An FFM, derived from the core design source, accurately models the core hardware's behavior. The design source may be Verilog or VHDL and may be register-transfer level, gate level, or a mixture of the two. Because these core macros require IP protection, the design source must be en-

INTEL-1049
U.S. Patent 7,151,430

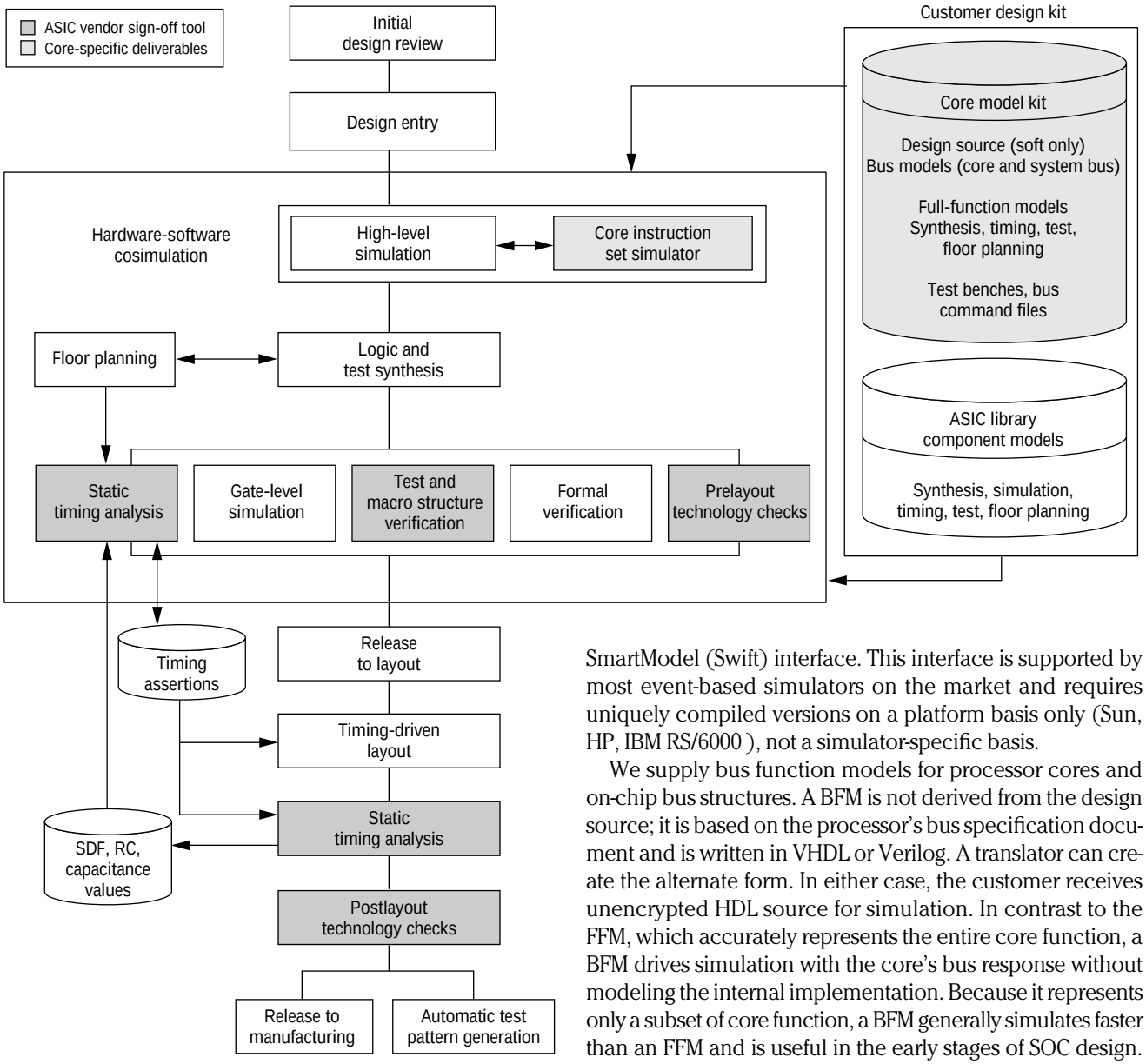

Figure 6. *Core-based ASIC design methodology.*

crypted but still function in simulation.

We protect the hard- and firm-core IP by compiling the model to a non-human-readable form, using Viewlogic's Verilog Model Compiler (VMC). This model can simulate directly with Cadence's Verilog XL and Viewlogic's Verilog Compiled Simulator tool. Communication between the model and the simulators takes place via the Verilog programming language interface (PLI). To support use of the compiled model with VHDL simulators, we integrated the model output from the VMC with a set of PLI routines known as the Synopsys SmartModel (Swift) interface. This interface is supported by most event-based simulators on the market and requires uniquely compiled versions on a platform basis only (Sun, HP, IBM RS/6000), not a simulator-specific basis.

We supply bus function models for processor cores and on-chip bus structures. A BFM is not derived from the design source; it is based on the processor's bus specification document and is written in VHDL or Verilog. A translator can create the alternate form. In either case, the customer receives unencrypted HDL source for simulation. In contrast to the FFM, which accurately represents the entire core function, a BFM drives simulation with the core's bus response without modeling the internal implementation. Because it represents only a subset of core function, a BFM generally simulates faster than an FFM and is useful in the early stages of SOC design.

Synthesis

The customer receives a soft core as a technology-dependent VHDL or Verilog netlist mapped to the same ASIC library as the customer logic. Soft-core synthesis usually consists of instantiating the netlist directly into the customer's design. In some cases, a small portion of a soft core may be parameterizable.

Elements within soft-core netlists used in the Synopsys synthesis environment carry the company's "dont_touch" annotation to prevent changes to the soft-core design during synthesis optimization. This annotation preserves the function and performance guaranteed by IBM. A customer who chooses to change a soft core beyond the allowed parame-

INTEL-1049
U.S. Patent 7,151,430

terization must explicitly remove the don't_touch annotation and subsequently accepts ownership of the core's function and timing.

Hard and firm cores are modeled as black box library elements in synthesis and pass through the synthesis process unchanged. The PowerPC CPU core is an exception to this rule. A synthesizable logic macro called the test mode matrix (TMM), required to support functional testing of the CPU, accompanies the PowerPC black box model. The customer synthesizes this logic to elements in the target ASIC library and can optimize it for both area and performance.

Timing

The customer performs timing analysis on soft cores using the timing models for the ASIC library elements. Timing assertions that specify false or don't-care paths in the design come with the soft-core netlist. The customer incorporates these assertions into the chip-level assertions used for timing sign-off.

The basic criterion for a soft core is that it meet performance requirements using the standard timing-driven layout system. Therefore, IBM provides core-specific wire-load models or area constraints on an exception basis only. The customers who designed the chips shown in Figures 2 and 4 integrated the soft cores with their custom logic and performed chip-level timing analysis. IBM placed and routed these chips, without soft-core region constraints, using the timing-driven layout approach. In contrast, the RAMDAC soft core in Figure 1 required region constraints and early floor planning, in addition to timing-driven layout, to meet the 220-MHz performance target.

For hard and firm cores, core developers must create black box timing models or timing abstracts to protect the intellectual content. The timing model must contain values for all pin-to-pin paths as well as all appropriate timing checks such as setup, hold, and minimum pulse width. Although the timing model is an abstracted representation (that is, it does not contain detailed design data), it must be accurate enough for static timing sign-off. Because of core designs' complexity and time-to-market requirements, an automated method of creating these models is necessary to eliminate human error.

We create the hard-core timing models for static timing sign-off using the IBM Einstimer tool's design abstraction process. Originally developed to support efficient timing of hierarchical designs, we extended this capability for cores. We read a detailed design netlist, RC (resistance-capacitance) and capacitance files from layout, and the design timing assertions into Einstimer. The tool generates a pin-to-pin timing abstraction for the design and writes out the information in Delay Calculation Language (DCL).

We do not use the detailed netlist and the layout RC and capacitance files to generate firm-core timing rules because the firm-core layout varies with each implementation. Instead, we use the timing assertions to generate fixed pin-to-pin delays that exactly match the delays published in the core's functional specification. We capture these delays, with variable temperature and voltage factors, and the appropriate timing checks in DCL statements. We compile DCL statements for both hard and firm cores into a non-human-readable executable form, which is provided to the customer for static timing analysis. From the abstracted DCL timing model, an IBM program called gensyn uses Einstimer to create timing information in the Synopsys synthesis model and core timing wrappers for simulation back-annotation.

Testability

Soft and firm cores are designed to meet the same DFT requirements as the customer design. All soft and firm cores must pass through the TSV sign-off tool without generating errors or warnings. All core scan chains and test clocks must be connected correctly, and untestable faults are not allowed. An edge clock at the core boundary drives the core. A clock splitter element in the design splits the edge clock into the required master/slave clocks. The customer connects the edge clock, scan clocks, and scan chain inputs and outputs in the customer logic to the corresponding pins at the core boundary. Once the core is fully integrated into the customer design, the customer uses ASIC library component test models to check it again for DFT compliance at the chip level.

We take either an integrated or an isolated approach to testing hard-core macros, depending on how the core was designed. If the hard-core design complies with the DFT requirements, we can test the core with the same test methods used in the standard ASIC flow (Figure 7). IBM includes the full gate-level core model and automatically generates test patterns for the customer and core logic concurrently. To maintain IP protection for hard and firm cores, we send an encrypted model to the customer. A special cloaking feature in the sign-off TSV tool prevents core models from being viewed via the graphical interface. We used the integrated core test method on the customer designs containing the PCI core chiplets described earlier.

In isolated testing, we test the customer logic separately from the core using a complex set of multiplexing and gating logic, called an isolation matrix, or test mode matrix (Figure 8). Control signals put the core into a series of modes. The nontest or functional mode allows communication between the core and customer logic. In this mode, the isolation matrix logic is transparent to the core's functional use. In core test modes, the core is accessible via the chip I/Os, and the customer logic is fenced off and stable. We can apply functional patterns to the core using the chip

INTEL-1049
U.S. Patent 7,151,430

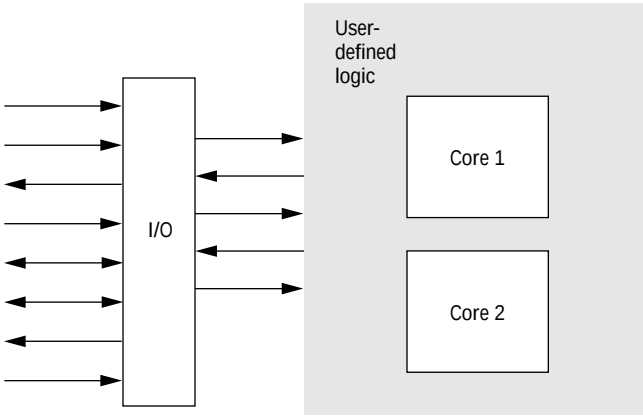

Figure 7. *Integrated core-testing approach.*

I/Os. We can also apply stored scan patterns directly to the core. In user-defined logic test mode, the customer logic is accessible via the chip I/Os, and the core is fenced off and stable. We test the customer logic by applying scan patterns using the chip I/Os.

Multiple cores on a single chip that require isolated testing will each need an associated isolation matrix. Adding this matrix may add several days to the design phase by increasing logic design and circuit wiring complexities.

We used the isolated test approach on the PowerPC CPU, the video DAC, and the Rambus cores. Analog cores often have additional unique test requirements, as was the case for the Rambus and video DAC. These cores must connect to specific, predetermined chip I/Os that are contacted by analog testers during manufacturing test.

Floor planning and layout

The customer performs floor planning of soft cores as part of chip-level design, using the ASIC library floor-planning models. It may be necessary to constrain the placement of soft-core logic, but the automatic placement tools often find a good solution without extraordinary intervention. RAMDAC, running at 220 MHz, was the only soft core that required advanced floor-planning and grouping restrictions to meet its performance target. We successfully placed, routed, and timed all the other soft cores in the designs described here using the timing-driven layout system with chip-level timing assertions.

IBM provides black box floor-plan models for each firm and hard core. These models accurately represent core size, aspect ratio, and pin locations. Firm cores resemble any other hierarchical block, except that processing them involves added complexity for the ASIC vendor, who must insert and remove the gate-level netlist. After inserting the netlist, we place and route each firm core and time it to meet the same timing assertions used to generate the static timing model.

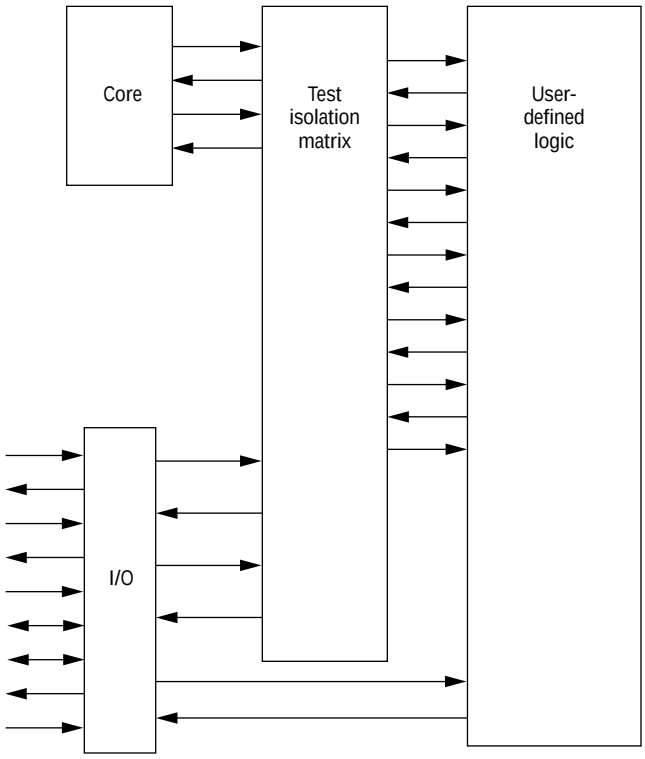

Figure 8. *Isolated core-testing approach.*

We remove the netlist information and replace it with the firm-core library element before returning the design to the customer for postlayout functional verification.

Hard cores complicate floor planning because of their size, wiring blockage, noise isolation requirements, peripheral test circuitry, and proximity to chip I/O pads. Although the PowerPC CPU core uses standard I/O cells and has no placement restrictions, it can use more than a quarter of a small chip's area. The core can be placed anywhere on the chip, but placement in a corner of the die allows for maximum wiring of the remaining chip logic. Corner placement also minimizes the chance of splitting a functional block and placing the parts on opposite sides of the core. We placed CPU pins that communicate with on-chip logic primarily on the north and east sides of the rectangular core structure. A small number of CPU pins—those that communicate with the chip I/Os—appear on the south and west sides of the core. If a critical timing path traverses the core, the designer can use region constraints to force logic to lie on one side of the core or block the area on one side to avoid the long wires.

Floor planning of chips with large hard cores is essential because of the wiring blockage these macros create. The PowerPC 401, the Rambus, and the PCI chiplets block all chip wiring tracks on the first two levels of metal. Porosity on

INTEL-1049
U.S. Patent 7,151,430

the third level of metal ranges from approximately 15% in the PCI chiplets to zero in the Rambus. In the 0.35-µm ASIC product, this leaves a minimum of one and a maximum of two wiring planes for routing over these cores. The limited porosity is caused primarily by the densely packed underlying core logic. Other core areas may complete the block wiring to prevent routing noise over the analog circuitry (in the Rambus) and instruction and data caches (in the 401 CPU). This causes highly congested wiring in other chip areas. It may be necessary to reserve area around the cores to help alleviate wiring congestion in those regions.

Certain hard cores, particularly those with analog functions, have additional characteristics affecting placement. The Rambus, for example, is a high-speed analog/digital memory controller with unique test and high-performance I/O cell requirements. The I/O cells must occupy predetermined locations on the die to allow access by wafer-level test equipment. The Rambus core requires wiring several sensitive high-current analog signals to the chip's I/O pads. Performance requirements on the signal nets between the core and the I/O pads and the need to prevent coupling to other noisy wires limit placement of the core: It must be placed in a predefined area directly adjacent to the test-specific I/O cells.

The chip shown in Figure 2 required both the 401 and Rambus cores to fit on a small die. Because the Rambus had to occupy a location in the middle of one side of the chip, the 401 was forced into the corner on the opposite side. This placement created a long narrow area between the two cores that caused significant chip-level routing challenges on the four-level metal design. The design was highly populated (using over 80% of the available silicon) and contained logic that needed to communicate across the core to other on-chip logic. Because routing across the core was available in only the vertical direction, we had to create additional reserve areas around the cores to allow for horizontal wires. Detailed floor planning and wiring congestion analysis highlighted these routing problems early in the design process. Subsequent versions of the 401 CPU contain wiring channels through the core on the third wiring level to help alleviate the chip-level routing problem.

Large cores sometimes require additional peripheral circuitry such as test logic, which must be arranged around the core. When test signals are multiplexed with functional signals around the chip's edge, the floor plan must account for the additional wire demand.

Logic and layout optimization of hard-core contents translates into area and performance improvements within the core. Inefficiencies at the chip level will partially offset these gains unless the cores and logic on the chip are arranged to avoid timing and wiring congestion problems.

With a good floor plan, laying out a chip with cores is much easier than laying out a chip without cores. For example, large static RAMs have been available as cores for many years, and these enable the layout engineer to add large amounts of memory to a chip without difficulty. Likewise, well-designed cores solve timing, clock skew, and congestion problems for high-performance circuits without additional layout effort on core contents.

Hardware-software cosimulation

Simulation and verification can quickly become the bottleneck in the design of large, complex core-based systems. A range of simulation models of varying accuracy, including BFMs and FFMs, address specific needs at various stages of the ASIC hardware design process. But none fulfills the needs of the software designer developing code for the embedded processor. Using an instruction set simulator (ISS) with an instruction set architecture (ISA) model has been a traditional method of software designers for code debug and execution time analysis. In stand-alone mode, an ISS runs several orders of magnitude faster than an FFM, executing an average of 100,000 instructions per second (IPS) versus the FFM's 5 to 20 IPS. An ISS also gives the software developer visibility into the internal registers of the processor as it executes instructions and provides breakpoint and single-step functions for controlling the execution stream.

Because the processor core is embedded on the same piece of silicon as the customer-designed ASIC logic, testing the interaction of processor and ASIC gates is critical. In most cases, the processor must correctly execute a stream of initialization code before it can begin to interact with the surrounding ASIC logic. We can debug this hardware-dependent software by translating processor instructions into a memory image and running an ASIC simulation with the FFM fetching instructions from the memory model. This method's productivity is constrained by the FFM's performance and the limited visibility this model provides into the processor's internals. As a result, many vendors are creating cosimulation products that link a processor ISS with an ASIC HDL simulator by using the processor BFM. These products execute processor instructions faster and concurrently simulate the activity of the remaining logic.

IBM used a prototype cosimulation system developed for the PowerPC 401 during the design of several core-based chips. The system consists of a 401 ISA model running in the PowerPC Virtual Simulator (PVS), linked with the VHDL simulator from Model Tech, Inc. (MTI), executing on an RS/6000 workstation. The ISA model does not include the concept of functional pins and uses the BFM to model output pin behavior. The BFM accepts bus commands such as read and write and translates them into bus transactions that model the interface's signal sequencing and timing.

The ISA model in PVS executes instructions at a relative-

INTEL-1049
U.S. Patent 7,151,430

ly high code throughput (much higher than an FFM would execute object code). It sends simple bus commands to the bus model in the MTI environment. It can also send commands to a file, so that a stand-alone BFM-based event-level simulation can run without requiring PVS at a later time. Designers debugging the software execution retain the visibility and instruction stream controls provided by PVS. The ASIC designer can debug problems detected in the ASIC logic in a familiar HDL simulator environment.

The design team for the chip in Figure 4 used the PVS/MTI cosimulation prototype to debug the instructions initializing the 401 processor and to configure ASIC logic functions on the PLB and OPB buses. Attempting a similar level of code debugging with the FFM model alone would have been virtually impossible. The cosimulation system provided visibility, control, and interaction with actual ASIC gates—essential to the team's understanding of processor functions and to the correct coding of chip initialization routines.

SOC DESIGN IS A PROCESS of system architecture design, block selection, integration, and verification. The designer is responsible for the integration of components and verification, as well as design of custom modules and modules not available in the core library. This shifts ASIC design to a block-based methodology that offers potentially huge productivity increases over RTL synthesis of the equivalent chip.

Critical methodology components include static sign-off using static timing analyzers, structured DFT schemes, synthesis, and floor planning. SOC design also requires new core models, hardware-software cosimulation techniques, consistent core development practices, and the availability of cores themselves. Cores must be designed for reuse. Reusable cores require a balance of completeness, flexibility, performance, and optimization. In the future, cores will consume most of the die area, so core evaluation and selection will be at least as significant as custom logic design. The SOC design methodology is still evolving. An important area of future research is the development of standards to foster core development, exchange, and interoperability.

Acknowledgments

We derived this article from two earlier works: "Core+ASIC Methodology: The Pursuit of System-on-a-Chip," A. Rincon et al., *Proc. Wescon IC Expo 97*, and "Design Environment for System-on-a-Chip," A. Rincon et al., *Proc. On-Chip System Conf., Design SuperCon*, 1997.

Ann Marie Rincon is a senior engineer in the ASIC Products Group of IBM Microelectronics in Essex Junction, Vermont. She is also the Core+ASIC Methodology Team leader. Rincon received a BS degree in mathematics and computer science from Saint Joseph's College in Indiana.

Cory Cherichetti is a staff engineer in the IBM ASIC Products Group, where he is working on the design of several cores and the supporting methodology. Cherichetti has a BSCSE degree from Rensselaer Polytechnical Institute.

James A. Monzel is a senior engineer in the IBM ASIC Products Group, where he is the Core+ASIC Test Methodology Team leader. He is the IEEE Computer Society's Test Technology chair for tutorials and the general chair of the North Atlantic Test Workshop. He is also on the steering committees of the VLSI Test Symposium and the Workshop on Testing of Embedded Core-Based Systems. Monzel received a BS from Case Western Reserve University and a BS in physics from Marietta College. He is a senior member of the IEEE.

David R. Stauffer is a senior engineer, responsible for core and ASIC design, in the IBM ASIC Products Group. Stauffer holds a BSEE degree from Pennsylvania State University and an MSEE degree from the University of Houston.

Michael T. Trick is a senior engineer in the IBM ASIC Products Group and the leader of the Physical Design Team for the ASIC Design Center. Trick received a BS degree in electrical engineering from the University of Illinois and a PhD in computer engineering from Carnegie Mellon University.

Address questions or comments about this article to Ann Marie Rincon, IBM Corp., 1000 River St., Dept. G07V, Bldg. 863-1, Essex Junction, VT 05452-4299; rincon@vnet.ibm.com.

INTEL-1049
U.S. Patent 7,151,430