2

Integrated Circuit Design Process and Electronic Design Automation

Robert Damiano, Raul Camposano, and Grant E. Martin

CONTENTS

2.1 INTRODUCTION

In this chapter, we describe the design process, its major stages, and how electronic design automation (EDA) tools fit into these processes. We also examine the interfaces between the major integrated circuit (IC) design stages as well as the kind of information—both abstractions upward and detailed design and verification information downward—that must flow between stages. We assume complementary metal oxide semiconductor (CMOS) is the basis for all technologies.

We will illustrate with a continuing example. A company wishes to create a new system on chip (SoC). The company assembles a product team, consisting of a project director, system architects, system verification engineers, circuit designers (both digital and analog), circuit verification engineers, layout engineers, and manufacturing process engineers. The product team determines the target technology and geometry as well as the fabrication facility or foundry. The system architects initially describe the system-level design (SLD) through a transaction-level specification in a language such as C++, SystemC, or Esterel. The system verification engineers determine the functional correctness of the SLD through simulation. The engineers validate the transaction processing through simulation vectors. They monitor the results for errors. Eventually, these same engineers would simulate the process with an identical set of vectors through the system implementation to see if the specification and the implementation match. There is some ongoing research to check this equivalence formally.

The product team partitions the SLD into functional units and hands these units to the circuit design teams. The circuit designers describe the functional intent through a high-level design language (HDL). The most popular HDLs are Verilog and VHDL. SystemVerilog is a recent language, adopted by the IEEE, which contains design, testbench, and assertion syntax. These languages allow the circuit designers to express the behavior of their design using high-level functions such as addition and multiplication. These languages allow expression of the logic at the register transfer level (RTL), in the sense that an assignment of registers expresses functionality. For the analog and analog mixed signal (AMS) parts of the design, there are also HDLs such as Verilog-AMS and VHDL-AMS. Most commonly, circuit designers use Simulation Program with IC Emphasis (SPICE) transistor models and netlists to describe analog components. However, high-level languages provide an easier interface between analog and digital segments of the design, and they allow writing higher-level behavior of the analog parts. Although the high-level approaches are useful as simulation model interfaces, there remains no clear method of synthesizing transistors from them. Therefore, transistor circuit designers usually depend on schematic capture tools to enter their data.

The design team must consider functional correctness, implementation closure (reaching the prioritized goals of the design), design cost, and manufacturability of a design. The product team takes into account risks and time to market as well as choosing the methodology. Anticipated sales volume can reflect directly on methodology; for instance, whether it is better to create a full-custom design, semicustom design or use standard cells, gate arrays, or a field programmable gate array (FPGA). Higher volume mitigates the higher cost of full-custom or semicustom design, while time to market might suggest using an FPGA methodology. If implementation closure for power and speed is tantamount, then an FPGA methodology might be a poor choice. Semicustom designs, depending on the required volume, can range from microprocessor central processor units (CPUs), digital signal processors (DSPs), application-specific standard parts (ASSP), or application-specific ICs (ASIC). In addition, for semicustom designs, the company needs to decide whether to allow the foundry to implement the layout or whether the design team should use customer-owned tooling (COT). We will assume that our product team chooses semicustom COT designs. We will mention FPGA and full-custom methodologies only in comparison.

In order to reduce cost, the product team may decide that the design warrants reuse of intellectual property (IP). IP reuse directly addresses the increasing complexity of design as opposed to feature geometry size. Reuse also focuses on attaining the goals of functional correctness. One analysis estimates that it takes 2000 engineering years and 1 trillion simulation vectors to verify 25 million lines of RTL code. Therefore, verified IP reuse reduces cost and time to market. Moreover, IP blocks themselves have become larger and more complex. For example, the 1176JZ-S ARM core is 24 times larger than the older 7TDI-S ARM core. The USB 2.0 Host is 23 times larger than the Universal Serial Bus (USB) 1.1 Device. PCI Express is 7.5 times larger than PCI v 1.1.

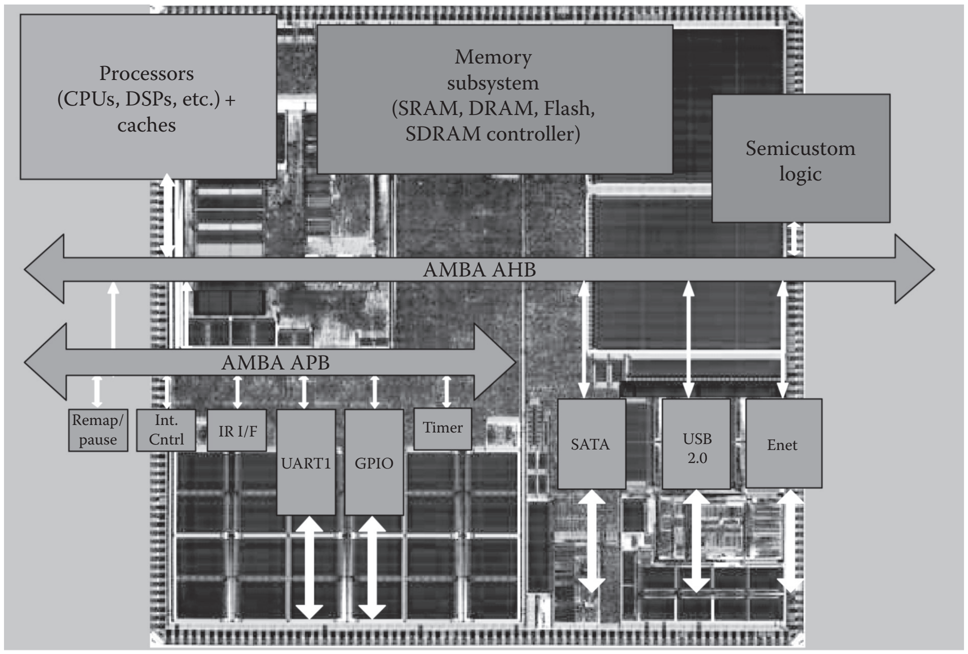
Processors (CPUs, DSPs, etc.) + caches
Memory subsystem (SRAM, DRAM, Flash, SDRAM controller)
Semicustom logic
AMBA AHB
AMBA APB
Remap/ pause
Int. Cntrl
IR I/F
UART1
GPIO
Timer
SATA
USB 2.0
Enet

FIGURE 2.1 SoC with IP.

Another important trend is that SoC-embedded memories are an increasingly large part of the SoC real estate. While in 1999 20% of a 180 nm SoC was embedded memory, roadmaps project that by 2005 embedded memory will consume 71% of a 90 nm SoC. These same roadmaps indicate that by 2014 embedded memory will grow to 94% of a 35 nm SoC.

SoCs typically contain one or more CPUs or DSPs (or both), a cache, a large amount of embedded memory, and many off-the-shelf components such as USB, universal asynchronous receiver–transmitter (UART), Serial Advanced Technology Attachment (SATA), and Ethernet (cf. Figure 2.1). The differentiating part of the SoC contains the new designed circuits in the product.

The traditional semicustom IC design flow typically comprises up to 50 steps. On the digital side of design, the main steps are functional verification, logical synthesis, design planning, physical implementation that includes clock-tree synthesis, placement and routing, extraction, design rule checking (DRC) and layout versus schematic (LVS) checking, static timing analysis, insertion of test structures, and test pattern generation. For analog designs, the major steps are as follows: schematic entry, SPICE simulation, layout, layout extraction, DRC, and LVS checking. SPICE simulations can include DC, AC, and transient analysis, as well as noise, sensitivity, and distortion analysis. Analysis and implementation of corrective procedures for the manufacturing process, such as mask synthesis and yield analysis, are critical at smaller geometries. In order to verify an SoC system where many components reuse IP, the IP provider may supply verification IPs, monitors, and checkers needed by system verification.

There are three basic areas where EDA tools assist the design team. Given a design, the first is verification of functional correctness. The second deals with implementation of the design. The last area deals with analysis and corrective procedures so that the design meets all manufacturability specifications. Verification, layout, and process engineers on the circuit design team essentially own these three steps.

2.2 VERIFICATION

The design team attempts to verify that the design under test (DUT) functions correctly. For RTL designs, verification engineers rely heavily on simulation at the cycle level. After layout, EDA tools, such as equivalence checking, can determine whether the implementation matches the RTL functionality. After layout, the design team must check that there are no

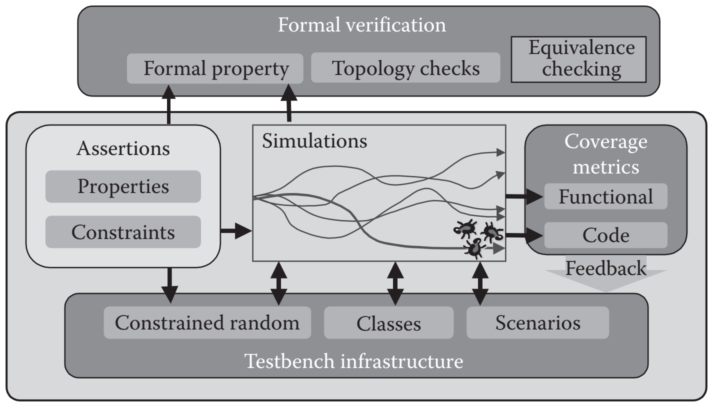

FIGURE 2.2 Digital simulation/formal verification.

problem delay paths. A static timing analysis tool can facilitate this. The team also needs to examine the circuit for noise and delay due to parasitics. In addition, the design must obey physical rules for wire spacing, width, and enclosure as well as various electrical rules. Finally, the design team needs to simulate and check the average and transient power. For transistor circuits, the design team uses SPICE circuit simulation or fast SPICE to determine correct functionality, noise, and power.

We first look at digital verification (cf. Figure 2.2). RTL simulation verifies that the DUT behavior meets the design intent. The verification engineers apply a set of vectors, called a testbench, to the design through an event-driven simulator, and compare the results to a set of expected outputs. The quality of the verification depends on the quality of the testbench. Many design teams create their testbench by supplying a list of the vectors, a technique called directed test. For a directed test to be effective, the design team must know beforehand what vectors might uncover bugs. This is extremely difficult since complex sequences of vectors are necessary to find some corner case errors. Therefore, many verification engineers create testbenches that supply stimulus through random vectors with biased inputs, such as the clock or reset signal. The biasing increases or decreases the probability of a signal going high or low. While a purely random testbench is easy to create, it suffers from the fact that vectors may be illegal as stimulus. For better precision and wider coverage, the verification engineer may choose to write a constrained random testbench. Here, the design team supplies random input vectors that obey a set of constraints.

The verification engineer checks that the simulated behavior does not have any discrepancies from the expected behavior. If the engineer discovers a discrepancy, then the circuit designer modifies the HDL and the verification engineer resimulates the DUT. Since exhaustive simulation is usually impossible, the design team needs a metric to determine quality. One such metric is coverage. Coverage analysis considers how well the test cases stimulate the design. The design team might measure coverage in terms of number of lines of RTL code exercised, whether the test cases take each leg of each decision, or how many "reachable" states are encountered.

Another important technique is for the circuit designer to add assertions within the HDL. These assertions monitor whether the internal behavior of the circuit is acting properly. Some designers embed tens of thousands of assertions into their HDL. Languages like SystemVerilog have extensive assertion syntax based on linear temporal logic. Even for languages without the benefit of assertion syntax, tool providers supply an application program interface (API), which allows the design team to build and attach its own monitors.

The verification engineer needs to run a large amount of simulation, which would be impractical if not for compute farms. Here, the company may deploy thousands of machines, 24/7, to enable the designer to get billions of cycles a day; sometimes the machines may run as many as 200 billion cycles a day. Best design practices typically create a highly productive computing environment. One way to increase throughput is to run a cycle simulation by taking a subset of the chosen verification language that both is synchronous and has a set of registers with clear clock cycles. This type of simulation assumes a uniformity of events and typically uses a time wheel with gates scheduled in a breadth-first manner.

Another way to tackle the large number of simulation vectors during system verification is through emulation or hardware acceleration. These techniques use specially configured hardware to run the simulation. In the case of hardware acceleration, the company can purchase special-purpose hardware, while in the case of emulation the verification engineer uses specially configured FPGA technology. In both cases, the system verification engineer must synthesize the design and testbench down to a gate-level model. Tools are available to synthesize and schedule gates for the hardware accelerator. In the case of an FPGA emulation system, tools can map and partition the gates for the hardware.

Of course, since simulation uses vectors, it is usually a less than exhaustive approach. The verification engineer can make the process complete by using assertions and formal property checking. Here, the engineer tries to prove that an assertion is true or to produce a counterexample. The trade-off is simple. Simulation is fast but by definition incomplete, while formal property checking is complete but may be very slow. Usually, the verification engineer runs constrained random simulation to unearth errors early in the verification process. The engineer applies property checking to corner case situations that can be extremely hard for the testbench to find. The combination of simulation and formal property checking is very powerful. The two can even be intermixed, by allowing simulation to proceed for a set number of cycles and then exhaustively looking for an error for a different number of cycles. In a recent design, by using this hybrid approach, a verification engineer found an error 21,000 clock cycles from an initial state. Typically, formal verification works well on specific functional units of the design. Between the units, the system engineers use an "assume/guarantee" methodology to establish block pre- and postconditions for system correctness.

During the implementation flow, the verification engineer applies equivalence checking to determine whether the DUT preserves functional behavior. Note that functional behavior is different from functional intent. The verification engineer needs RTL verification to compare functional behavior with functional intent. Equivalence checking is usually very fast and is a formal verification technology, which is exhaustive in its analysis. Formal methods do not use vectors.

For transistor-level circuits, such as analog, memory, and radio frequency, the event-driven verification techniques suggested earlier do not suffice (cf. Figure 2.3). The design team needs to compute signals accurately through SPICE circuit simulation. SPICE simulation is very time consuming because the algorithm solves a system of differential equations. One way to get around this cost is to select only a subset of transistors, perform an extraction of the parasitics, and then simulate the subset with SPICE. This reduction gives very accurate results for the subset, but even so, the throughput is still rather low. Another approach is to perform a fast SPICE simulation.

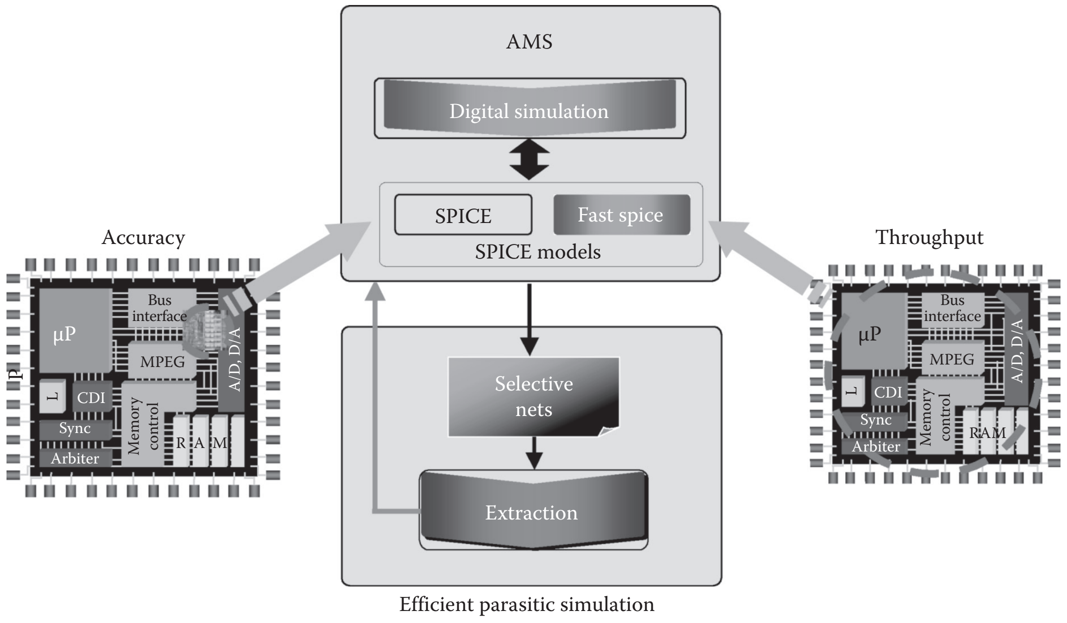

FIGURE 2.3 Transistor simulation with parasitics.

This last SPICE approach trades some accuracy for a significant increase in throughput. The design team can also perform design space exploration by simulating various constraint values on key goals such as gain or phase margin to find relatively optimal design parameters. The team analyzes the multiple-circuit solutions and considers the cost trade-offs. A new generation of tools performs this "design exploration" in an automatic manner. Mixed-level simulation typically combines RTL, gate, and transistor parts of the design and uses a communication backplane to run the various simulations and share input and output values.

Finally, for many SoCs, both hardware and software comprise the real system. System verification engineers may run a hardware–software cosimulation before handing the design to a foundry. All simulation system components mentioned can be part of this cosimulation. In early design stages, when the hardware is not ready, the software can simulate (*execute*) an instruction set model, a virtual prototype (model), or an early hardware prototype typically implemented in FPGAs.

2.3 IMPLEMENTATION

This brings us to the next stage of the design process, the implementation and layout of the digital design. Circuit designers implement analog designs by hand. FPGA technologies usually have a single basic combinational cell, which can form a variety of functions by constraining inputs. Layout and process tools are usually proprietary to the FPGA family and manufacturer. For semicustom design, the manufacturer supplies a precharacterized cell library, either standard cell or gate array. In fact, for a given technology, the foundry may supply several libraries, differing in power, timing, or yield. The company decides on one or more of these as the target technology. One twist on the semicustom methodology is structured ASIC. Here, a foundry supplies preplaced memories, pad rings, and power grids, as well as sometimes preplaced gate array logic, which is similar to the methodology employed by FPGA families. The company can use semicustom techniques for the remaining combinational and sequential logic. The goal is to reduce nonrecurring expenses by limiting the number of mask sets needed and by simplifying physical design.

By way of contrast, in a full-custom methodology, one tries to gain performance and limit power consumption by designing much of the circuit as transistors. The circuit designers keep a corresponding RTL design. The verification engineer simulates the RTL and extracts a netlist from the transistor description. Equivalence checking compares the extracted netlist to the RTL. The circuit designer manually places and routes the transistor-level designs. Complex high-speed designs, such as microprocessors, sometimes use full custom methodology, but the design costs are very high. The company assumes that the high volume will amortize the increased cost. Full-custom designs consider implementation closure for power and speed as most important. At the other end of the spectrum, FPGA designs focus on design cost and time to market. Semicustom methodology tries to balance the goals of timing and power closure with design cost (cf. Figure 2.4).

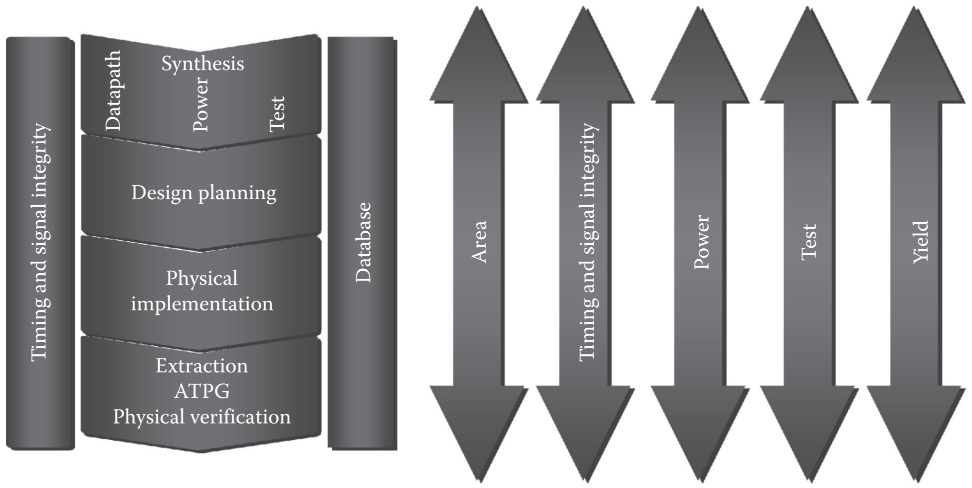

FIGURE 2.4 Multi-objective implementation convergence.

In the semicustom implementation flow, one first attempts to synthesize the RTL design into a mapped netlist. The circuit designers supply their RTL circuit along with timing constraints. The timing constraints consist of signal arrival and slew (transition) times at the inputs and required times and loads (capacitances) at the outputs. The circuit designer identifies clocks as well as any false or multiple-cycle paths. The technology library is usually a file that contains a description of the function of each cell along with delay, power, and area information. The cell description contains the pin-to-pin delay represented either as lookup table functions of input slew, output load, and other physical parameters, such as voltage and temperature, or as polynomial functions that best fit the parameter data. For example, foundries provide cell libraries in Liberty or OLA (Open Library Application Programming Interface) formats. The foundry also provides a wire delay model, derived statistically from previous designs. The wire delay model correlates the number of sinks of a net to capacitance and delay.

Several substages comprise the operation of a synthesis tool. First, the synthesis tool compiles the RTL into technology-independent cells and then optimizes the netlist for area, power, and delay. The tool maps the netlist into a technology. Sometimes, synthesis finds complex functions such as multipliers and adders in parameterized (area/timing) reuse libraries. For example, the tool might select a Booth multiplier from the reuse library to improve timing. For semicustom designs, the foundry provides a standard cell or gate array library, which describes each functional member. In contrast, the FPGA supplier describes a basic combinational cell from which the technology mapping matches functional behavior of subsections of the design. To provide correct functionality, the tool may set several pins on the complex gates to constants. A postprocess might combine these functions for timing, power, or area.

A final substage tries to analyze the circuit and performs local optimizations that help the design meet its timing, area, and power goals. Note that due to finite number of power levels of any one cell, there are limits to the amount of capacitance that functional cell types can drive without the use of buffers. Similar restrictions apply to input slew (transition delay). The layout engineer can direct the synthesis tool by enhancing or omitting any of these stages through scripted commands. Of course, the output must be a mapped netlist.

To get better timing results, foundries continue to increase the number of power variations for some cell types. One limitation to timing analysis early in the flow is that the wire delay models are statistical estimates of the real design. Frequently, these wire delays can differ significantly from those found after routing. One interesting approach to synthesis is to extend each cell of the technology library so that it has an infinite or continuous variation of power. This approach, called gain-based synthesis, attempts to minimize the issue of inaccurate wire delay by assuming cells can drive any wire capacitance through appropriate power level selection. In theory, there is minimal perturbation to the natural delay (or gain) of the cell. This technique makes assumptions such as that the delay of a signal is a function of capacitance. This is not true for long wires where resistance of the signal becomes a factor. In addition, the basic approach needs to include modifications for slew (transition delay).

To allow detection of manufacturing faults, the design team may add extra test generation circuitry. Design for test (DFT) is the name given to the process of adding this extra logic (cf. Figure 2.5). Sometimes, the foundry supplies special registers, called logic-sensitive scan devices. At other times, the test tool adds extra logic called Joint Test Action Group (JTAG) boundary scan logic that feeds the registers. Later in the implementation process, the design team will generate data called scan vectors that test equipment uses to detect manufacturing faults. Subsequently, tools will transfer these data to automatic test equipment (ATE), which perform the chip tests.

As designs have become larger, so has the amount of test data. The economics of the scan vector production with minimal cost and design impact leads to data compression techniques. One of the most widely used techniques is deterministic logic built-in self-test (BIST). Here, a test tool adds extra logic on top of the DFT to generate scan vectors dynamically.

Before continuing the layout, the engineer needs new sets of rules, dealing with the legal placement and routing of the netlist. These libraries, in various exchange formats, for example, LEF for logic, DEF for design, and PDEF for physical design, provide the layout engineer physical directions and constraints. Unlike the technology rules for synthesis, these rules are typically model dependent. For example, there may be information supplied by the circuit designer

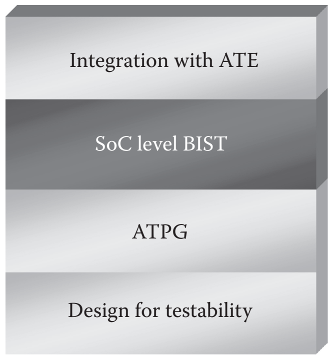

FIGURE 2.5 Design for test.

about the placement of macros such as memories. The routing tool views these macros as blockages. The rules also contain information from the foundry.

Even if the synthesis tool preserved the original hierarchy of the design, the next stages of implementation need to view the design as flat. The design-planning step first flattens the logic and then partitions the flat netlist as to assist placement and routing; in fact, in the past, design planning was sometimes known as floor planning. A commonly used technique is for the design team to provide a utilization ratio to the design planner. The utilization ratio is the percentage of chip area used by the cells as opposed to the nets. If the estimate is too high, then routing congestion may become a problem. If the estimate is too low, then the layout could waste area. The design-planning tool takes the locations of hard macros into account. These macros are hard in the sense that they are rectangular with a fixed length, fixed width, and sometimes a fixed location on the chip. The design-planning tool also tries to use the logical hierarchy of the design as a guide to the partitioning. The tool creates, places, and routes a set of macros that have fixed lengths, widths, and locations. The tool calculates timing constraints for each macro and routes the power and ground grids. The power and ground grids are usually on the chip's top levels of metal and then distributed to the lower levels. The design team can override these defaults and indicate which metal layers should contain these grids. Sometimes design planning precedes synthesis. In these cases, the tool partitions the RTL design and automatically characterizes each of the macros with timing constraints.

After design planning, the layout engineer runs the physical implementation tools on each macro. First, the placer assigns physical locations to each gate of the macro. The placer typically moves gates while minimizing some cost, for example, wire length or timing. Legalization follows the coarse placement to make sure the placed objects fit physical design rules. At the end of placement, the layout engineer may run some more synthesis, like resizing of gates. One of the major improvements to placement over the last decade is the emergence of physical synthesis. In physical synthesis, the tool interleaves synthesis and placement. Recall that previously, logic synthesis used statistical wire capacitance. Once the tool places the gates, it can perform a global route and get capacitances that are more accurate for the wires, based on actual placed locations. The physical synthesis tool iterates this step and provides better timing and power estimates.

Next, the layout engineer runs a tool that buffers and routes the clock tree. Clock-tree synthesis attempts to minimize the delay while assuring that skew, that is the variation in signal transport time from the clock to its corresponding registers, is close to zero.

Routing the remaining nets comes after clock-tree synthesis. Routing starts with a global analysis called global route. Global route creates coarse routes for each signal and its outputs. Using the global routes as a guide, a detailed routing scheme, such as a maze channel or switchbox, performs the actual routing. As with the placement, the tool performs a final legalization to assure that the design obeys physical rules. One of the major obstacles to routing is signal congestion. Congestion occurs when there are too many wires competing for a limited amount of chip wire resource. Remember that the design team gave the design planner a utilization ratio in the hope of avoiding this problem.

Both global routing and detailed routing take the multilayers of the chip into consideration. For example, the router assumes that the gates are on the polysilicon layer, while the wires connect the gates through vias on 3–8 layers of metal. Horizontal or vertical line segments comprise the routes, but some recent work allows 45° lines for some foundries. As with placement, there may be some resynthesis, such as gate resizing, at the end of the detailed routing stage.

Once the router finishes, an extraction tool derives the capacitances, resistances, and inductances. In a 2D parasitic extraction, the extraction tool ignores 3D details and assumes that each chip level is uniform in one direction. This produces only approximate results. In the case of the much slower 3D parasitic extraction, the tool uses 3D field solvers to derive very accurate results. A 2 1/2-D extraction tool compromises between speed and accuracy. By using multiple passes, it can access some of the 3D features. The extraction tool places its results in a standard parasitic exchange format file.

During the implementation process, the verification engineer continues to monitor behavioral consistency through equivalence checking and using LVS comparison. The layout engineer analyzes timing and signal integrity issues through timing analysis tools and uses their results to drive implementation decisions. At the end of the layout, the design team has accurate resistances, capacitances, and inductances for the layout. The system engineer uses a sign-off timing analysis tool to determine if the layout meets timing goals. The layout engineer needs to run a DRC on the layout to check for violations.

Both the Graphic Data System II (GDSII) and the Open Artwork System Interchange Standard (OASIS) are databases for shape information to store a layout. While the older GDSII was the database of choice for shape information, there is a clear movement to replace it by the newer, more efficient OASIS database. The LVS tool checks for any inconsistencies in this translation.

What makes the implementation process so difficult is that multiple objectives need consideration. For example, area, timing, power, reliability, test, and yield goals might and usually cause conflict with each other. The product team must prioritize these objectives and check for implementation closure.

Timing closure—that is meeting all timing requirements—by itself is becoming increasingly difficult and offers some profound challenges. As process geometry decreases, the significant delay shifts from the cells to the wires. Since a synthesis tool needs timing analysis as a guide and routing of the wires does not occur until after synthesis, we have a chicken and egg problem. In addition, the thresholds for noise sensitivity also shrink with smaller geometries. This along with increased coupling capacitances, increased current densities and sensitivity to inductance, make problems like crosstalk and voltage (IR) drop increasingly familiar.

Since most timing analysis deals with worst-case behavior, statistical variation and its effect on yield add to the puzzle. Typically timing analysis computes its cell delay as function of input slew (transition delay) and output load (output capacitance or RC). If we add the effects of voltage and temperature variations as well as circuit metal densities, timing analysis gets to be very complex. Moreover, worst-case behavior may not correlate well with what occurs empirically when the foundry produces the chips. To get a better predictor of parametric yield, some layout engineers use statistical timing analysis. Here, rather than using single numbers (worst case, best case, corner case, nominal) for the delay-equation inputs, the timing analysis tool selects probability distributions representing input slew, output load, temperature, and voltage among others. The delay itself becomes a probability distribution. The goal is to compute the timing more accurately in order to create circuits with smaller area and lower power but with similar timing yield.

Reliability is also an important issue with smaller geometries. Signal integrity deals with analyzing what were secondary effects in larger geometries. These effects can produce erratic behavior for chips manufactured in smaller geometries. Issues such as crosstalk, IR drop, and electromigration are factors that the design team must consider in order to produce circuits that perform correctly.

Crosstalk noise can occur when two wires are close to each other (cf. Figure 2.6). One wire, the aggressor, switches while the victim signal is in a quiet state or making an opposite transition. In this case, the aggressor can force the victim to glitch. This can cause a functional failure or can simply consume additional power. Gate switching draws current from the power and

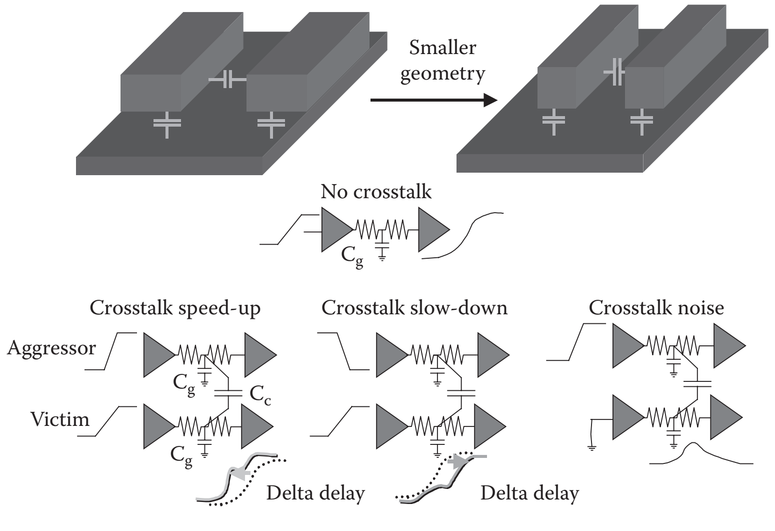

FIGURE 2.6 Crosstalk.

ground grids. That current, together with the wire resistance in the grids, can cause significant fluctuations in the power and ground voltages supplied to gates. This problem, called IR drop, can lead to unpredictable functional errors. Very high frequencies can produce high current densities in signals and power lines, which can lead to the migration of metal ions. This power electromigration can lead to open or shorted circuits and subsequent signal failure.

Power considerations are equally complex. As the size of designs grows and geometries shrink, power increases. This can cause problems for batteries in wireless and handheld devices and thermal management in microprocessor, graphic, and networking applications. Power consumption falls into two areas: (1) dynamic power (cf. Figure 2.7), the power consumed when devices switch value, and (2) leakage power (cf. Figure 2.8), the power leaked through the transistor. Dynamic power consumption grows directly with increased capacitance and voltage. Therefore, as designs become larger, dynamic power increases. One easy way to reduce dynamic power is to decrease voltage. However, decreased voltage leads to smaller noise margins and less speed.

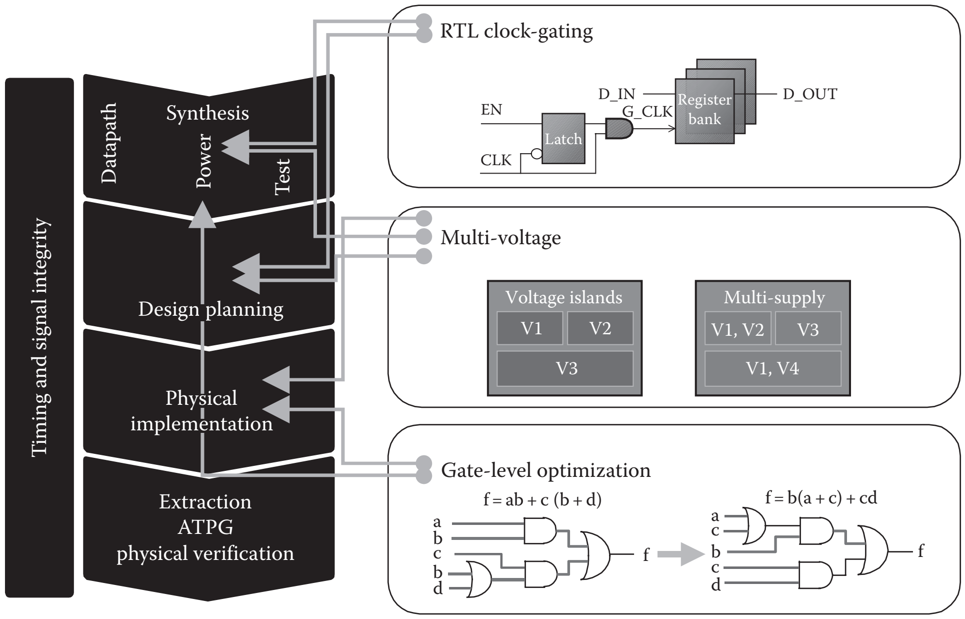

FIGURE 2.7 Dynamic power management.

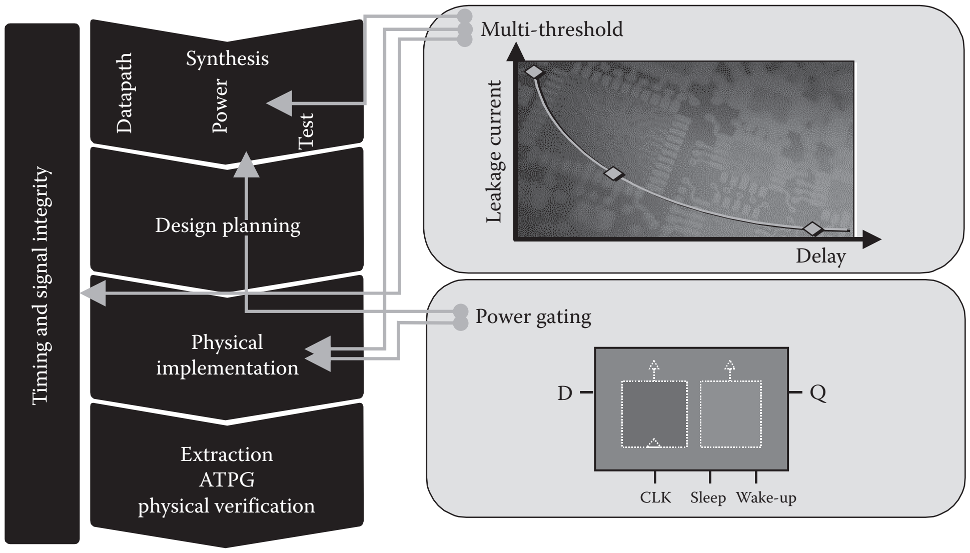

FIGURE 2.8 Static power management (leakage).

A series of novel design and transistor innovations can reduce the power consumption. These include operand isolation, clock gating, and voltage islands. Timing and power considerations are very often in conflict with each other, so the design team must employ these remedies carefully.

A design can have part of its logic clock gated by using logic to enable the bank of registers. The logic driven by the registers is quiescent until the clock-gated logic enables the registers. Latches at the input can isolate parts of a design that implement operations (e.g., an arithmetic logic unit), when results are unnecessary for correct functionality, thus preventing unnecessary switching. Voltage islands help resolve the timing versus power conflicts. If part of a design is timing critical, a higher voltage can reduce the delay. By partitioning the design into voltage islands, one can use lower voltage in all but the most timing-critical parts of the design. An interesting further development is dynamic voltage/frequency scaling, which consists of scaling the supply voltage and the speed during operation to save power or increase performance temporarily.

The automatic generation of manufacturing fault detection tests was one of the first EDA tools. When a chip fails, the foundry wants to know why. Test tools produce scan vectors that can identify various manufacturing faults within the hardware. The design team translates the test vectors to standard test data format and the foundry can inject these inputs into the failed chip through automated test equipment (ATE). Remember that the design team added extra logic to the netlist before design planning, so that test equipment could quickly insert the scan vectors, including set values for registers, into the chip. The most common check is for stuck at 0 or stuck at 1 faults where the circuit has an open or short at a particular cell. It is not surprising that smaller geometries call for more fault detection tests. An integration of static timing analysis with transition/path delay fault automatic test pattern generation (ATPG) can help, for example, to detect contact defects; while extraction information and bridging fault ATPG can detect metal defects.

Finally, the design team should consider yield goals. Manufacturing becomes more difficult as geometries shrink. For example, thermal stress may create voids in vias. One technique to get around this problem is to minimize the vias inserted during routing, and for those inserted, to create redundant vias. Via doubling, which converts a single via into multiple vias, can reduce resistance and produce better yield. Yield analysis can also suggest wire spreading during routing to reduce crosstalk and increase yield. Manufacturers also add a variety of manufacturing process rules needed to guarantee good yield. These rules involve antenna checking and repair through diode insertion as well as metal fill needed to produce uniform metal densities necessary for copper wiring chemical–mechanical polishing (CMP). Antenna repair has little to do with what we typically view as antennas. During the ion-etching process, charge collects on the wires connected to the polysilicon gates. These charges can damage the gates. The layout tool can connect small diodes to the interconnect wires as a discharge path.

Even with all the available commercial tools, there are times when layout engineers want to create their own tool for analysis or small implementation changes. This is analogous to the need for an API in verification. Scripting language and C-language-based APIs for design databases such as MilkyWay and OpenAccess are available. These databases supply the user with an avenue to both the design and rules. The engineer can directly change and analyze the layout.

2.4 DESIGN FOR MANUFACTURING

One of the newest areas for EDA tools is design for manufacturing. As in other areas, the driving force of the complexity is the shrinking of geometries. After the design team translates their design to shapes, the foundry must transfer those shapes to a set of masks. Electron beam (laser) equipment then creates the physical masks for each layer of the chip from the mask information. For each layer of the chip, the foundry applies photoresistive material and then transfers the mask structures by the stepper optical equipment onto the chip. Finally, the foundry etches the correct shapes by removing the excess photoresist material.

Since the stepper uses light for printing, it is important that the wavelength is small enough to transcribe the features accurately. When the chip's feature size was 250 nm, we could use lithography equipment that produced light at a wavelength of 248 nm. New lithography equipment that

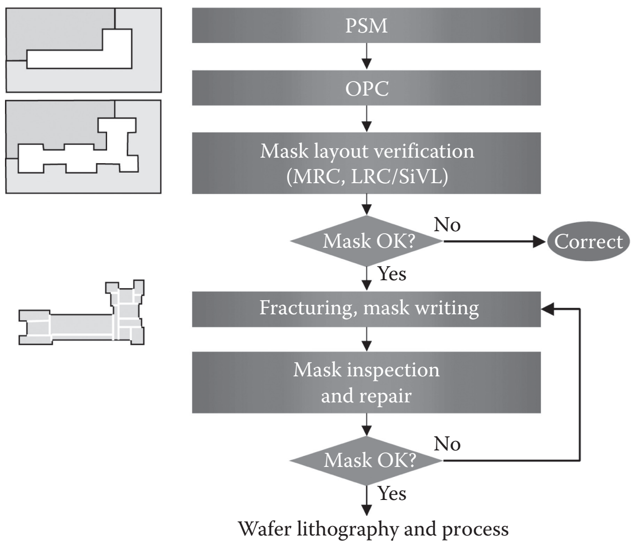

FIGURE 2.9 Subwavelength: from layout to masks.

produces light of lower wavelength needs significant innovation and can be very expensive. When the feature geometry gets significantly smaller than the wavelength, the detail of the reticles (fine lines and wires) transferred to the chip from the mask can be lost. EDA tools can analyze and correct this transfer operation without new equipment, by modifying the shapes of data—a process known as mask synthesis (cf. Figure 2.9). This process uses resolution enhancement techniques and methods to provide dimensional accuracy.

One mask synthesis technique is optimal proximity correction (OPC). This process takes the reticles in the GDSII or OASIS databases and modifies them by adding new lines and wires, so that even if the geometry is smaller than the wavelength, optical equipment adequately preserves the details. This technique successfully transfers geometric features of down to one-half of the wavelength of the light used. Of course given a fixed wavelength, there are limits beyond which the geometric feature size is too small for even these tricks.

For geometries of 90 nm and below, the lithography EDA tools combine OPC with other mask synthesis approaches such as phase shift mask (PSM), off-axis illumination, and assist features (AF). For example, PSM is a technique where the optical equipment images dark features at critical dimensions with 0° illumination on one side and 180° illumination on the other side. There are additional manufacturing process rules needed, such as minimal spacing and cyclic conflict avoidance, to avoid situations where the tool cannot map the phase.

In summary, lithography tools proceed through PSM, OPC, and AF to enhance resolution and make the mask more resistive to process variations. The process engineer can perform a verification of silicon versus layout and a check of lithography rule compliance. If either fails, the engineer must investigate and correct, sometimes manually. If both succeed, another EDA tool "fractures" the design, subdividing the shapes into rectangles (trapezoids), which can be fed to the mask writing equipment. The engineer can then transfer the final shapes file to a database, such as the manufacturing-electron-beam-exposure system (MEBES). Foundry equipment uses the MEBES database (or other proprietary formats) to create the physical masks. The process engineer can also run a *virtual* stepper tool to preanalyze the various stages of the stepper operation. After the foundry manufactures the masks, a mask inspection and repair step ensures that they conform to manufacturing standards.

Another area of design for manufacturing analysis is the prediction of yield (cf. Figure 2.10). The design team would like to correlate some of the activities during route with actual yield. Problems with CMP via voids and crosstalk can cause chips to unexpectedly fail. EDA routing tools offer some solutions in the form of metal fill, via doubling and wire spacing. Library providers are starting to develop libraries for higher yields that take into account several yield failure mechanisms. There are tools that attempt to correlate these solutions with yield. Statistical timing analysis can correlate timing constraints to parametric circuit yield.

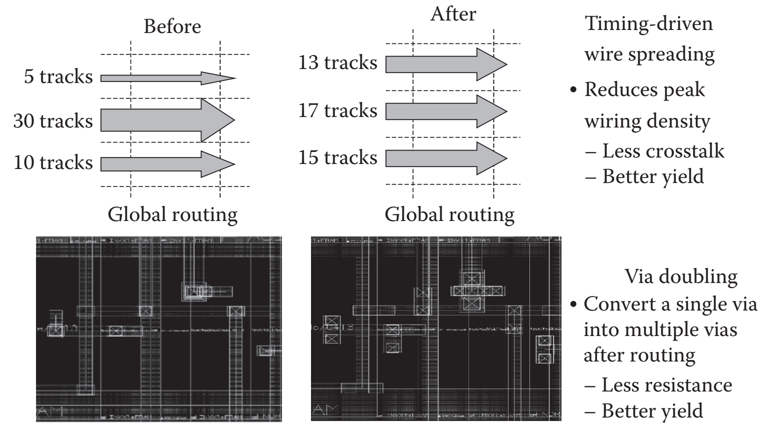

FIGURE 2.10 Yield enhancement features in routing.

Finally, the process engineer can use tools to predict the behavior of transistor devices or processes. Technology computer-aided design (TCAD) deals with the modeling and simulation of physical manufacturing process and devices. Engineers can model and simulate individual steps in the fabrication process. Likewise, the engineer can model and simulate devices, parasitics, or electrical/thermal properties, therefore providing insights into their electrical, magnetic, or optical properties.

For example, because of packing density, foundries may switch isolation technology for an IC from the local oxidation of silicon model toward the shallow trench isolation (STI) model. Under this model, the process engineer can analyze breakdown stress, electrical behavior such as leakage, or material versus process dependencies. TCAD tools can simulate STI effects; extract interconnect parasitics, such as diffusion distance, and determine SPICE parameters.

2.5 UPDATE: A DECADE OF EVOLUTION IN DESIGN PROCESS AND EDA

Reviewing the design process described in this chapter, which was written for the first edition of the handbook published in 2006, it is remarkable how similar today's IC design process is in terms of major tasks and steps, despite the decade that has elapsed, and the huge increase in SoC complexity and process complexity that has occurred. In addition, despite many changes in the EDA industry and its tools, there are remarkable similarities in many ways.

Nevertheless, the past decade has seen some important changes that are worthy of note for their impact on design process and EDA tools. Some of these arguably have more impact in the economics of design than on the technical design processes. In the following update, we try to briefly note some of these major changes and summarize their impact on design process and EDA:

- *SoC design complexity*: From a few tens of blocks and fewer than 10 subsystems in a complex SoC of a decade ago to today's most complex SoCs that may have hundreds of blocks and many complicated subsystems, we have seen the design tasks grow in magnitude. This has many side effects: a large increase in design teams, greater collaboration by design teams around the world, many mergers, acquisitions and restructurings of teams in design companies (especially to acquire design IP and design talent in subsystem areas that are being merged into the more complex SoCs), and much more reliance on third-party IP blocks and subsystems. In addition, less and less can be done by last-minute manual tweaks to the design, hence resulting in more reliance on EDA tools, quality IP, and very well-controlled design processes. The latest *International Technology Roadmap for Semiconductors* (2012 update edition) [27] has substantial discussions of SoC design challenges in its "System Drivers" and "Design" chapters.
- *Reduction in the number of semiconductor fabrication facilities*: The trend, which started in the 1990s and accelerated in the last decade, of system companies shedding their

in-house semiconductor fabrication facilities, and even some semiconductor companies reducing their dependence on in-house fabrication, has led to significant changes in design practices. Fabrication processes have become more standardized with a shrink in the number of potential suppliers; the issue of *second sourcing* has changed its form. It no longer relies on second source fabrication of a single design; rather, these days, systems companies change chip sets for standard functions such as modems or application processors, either by accepting complete systems from providers (e.g., modems) or by using software layers to differentiate their products and recompile it on new application processors. Thus, differentiation is more by software than hardware design. As processes moved down in scale, fewer and fewer companies could afford to stay on the fabrication process treadmill. Several consortia were formed to develop common processes, but this eventually became unaffordable for all but a few companies.

- *Growing importance of fabless semiconductor industry*: As system companies switched away from in-house process development and fabrication facilities, we have seen the rise of the fabless semiconductor industry and the pure-play foundries, the most notable being TSMC [28]. This has kept alive the dream of ASIC design in system companies, fostered an intermediate fabless semiconductor network of companies (such as Qualcomm, Broadcom, and many more), allowed some semiconductor companies to run a mixed fab/fabless business model, and standardized design processes, which helps design teams deal with the complexity of design in advanced processes at 28 nm and below. It has also fostered the growth of companies, such as eSilicon, that act as intermediaries between fabless IC companies that focus on RTL and above and fabrication. This involves essentially driving the back-end tools, developing a special relationship with fabrication that ensures faster access to technology information, and shielding fabless IC companies from the more intimate knowledge of technology. Keeping this separate from the IP business has also helped reduce the fabless industry concerns about design services.
- *Consolidation in the EDA industry*: It has become increasingly difficult to fund EDA startups, and the "Big 3" of EDA (Synopsis, Cadence, and Mentor Graphics) have continued buying many of the small startups left in the industry over the last decade, including Forte, Jasper, Springsoft, EVE, and Nimbic [29]. For Cadence and Synopsis, substantial recent mergers and acquisitions include considerable amounts of design IP and IP design teams. This consolidation has good points and bad points in terms of the design process and tools and flows. On the bad side, it may limit some of the innovation that small EDA startups bring to the industry. On the good side, the larger EDA companies have more to invest in new tools and capabilities and flows, are able to work more closely with large IP providers and the various fabless and integrated (with fabrication facilities) semiconductor companies, and can make strategic investments to match tool capabilities to future process generations. In addition, by branching out to include IP as part of the EDA business model, the tools can be improved with in-house designs and the IP offerings themselves have the backing of large EDA companies and their strategic investments. This tends to increase the variety and quality of the IP available to design teams and offers them more choices.
- *Relative geographic shifts in design activity*: Over the years there has been a tremendous reorganization of design activity. In a geographic sense, design activity in Europe and North America has declined (Europe more than North America), Japan has declined to a lesser extent, and the rest of Asia—Taiwan, Korea, and China—has increased. In India, design activity has also increased. This has had some effect on the adoption and deployment of advanced design methodologies and tools. In particular, electronic system-level (ESL) tools and methods have been of greater interest in Europe and to some extent in North America. This is fostered in Europe by EU programs that fund advanced work in microelectronics and design. The geographic shift has meant that ESL methodologies have had slower adoption than might have been the case if European design activity had remained as strong. However, we have seen more interest in the last few years in the use of system-level methods and models in Asian design companies, ameliorating some of this delay. Japan may be an exception to this, with early and continued interest in ESL.

- *Advanced processes*: The development in process technologies—from 90 nm through 65 nm, 40/45 nm, 28/30/32 nm, 20/22 nm, to 14/16 nm, including new circuit designs such as FinFETs—has continued apace over the last decade, albeit at a somewhat slower pace than traditional Moore's law would indicate (perhaps taking 3 years rather than 2 years to double transistor count). Inherent speeds of transistors no longer increase at the old pace. Some reports claim that even the cost per transistor has reached its minimum value around the 32–22 nm nodes. If true, this would mean that functionality no longer becomes cheaper with every generation and that a completely different strategy would be required in the whole electronic industry. More advanced design techniques are required. Design teams need to grow larger and integrate more diverse subsystems into complex SoCs. Design closure becomes harder and the number of steps required at each phase of design grows.
- *Changes in SoC architectures and more platforms*: The prediction made in 1999 of the move to a platform-based design style for SoCs [30] has come true; almost all complex SoCs follow some kind of platform-based design style in which most new products are controlled derivatives of the platform. This has shifted the hardware/software divide toward relatively fewer custom hardware blocks (only where their performance, power, or area characteristics are vital to the product function) and relatively more processors (whether general purpose or application specific) and memory (for all that software and associated data).
- *Greater use of FPGAs versus ASICs*: Although FPGA vendors have long proclaimed the near-complete takeover of FPGAs from ASICs in electronics product design, it has not unfolded according to their optimistic predictions [31]. However, FPGAs are used more in end-product delivery these days than a decade ago, at least in some classes of products. The new FPGAs with hardened embedded processors and other hard elements such as multiply-accumulate DSP blocks and SERDES look more attractive as product development and delivery vehicles than pure FPGAs. A notable example of this is Xilinx's Zynq platform FPGAs [32]. Use of such design platforms changes aspects of the design process rather significantly.
- *Greater use of IP blocks*: The use of design IP blocks and subsystems, whether sourced from internal design groups or external IP providers, has grown significantly. While reducing new design effort on complex SoCs, the use of IP blocks also changes much of the focus of SoC from new original design to architecting the use of IP and the integration and verification process itself.
- *Greater use of processors and application-specific instruction set processors* (*ASIPs*): The kind of IP block has a big influence on SoC design. Increasingly, platforms are using more processors, both general-purpose control plane processors and dedicated DSPs, accelerators, and ASIPs in the dataplane. In some domains, such as audio decoding, ASIPs are so close in performance, energy consumption, and cost (area) to dedicated hardware blocks, in which their flexibility in supporting multiple standards and new standards makes their use almost obligatory. The shifting boundary between dedicated hardware blocks and processors moves the design boundary between hardware and software in general, thus changing in degree (but not in kind) the design effort devoted to the software side.
- *Growing reliance on software*: This is a natural outgrowth of the increased use of processors discussed earlier. And the shift of the hardware/software divide changes the emphasis in design teams. This handbook, particularly the design process in this chapter, emphasizes hardware design of SoCs, of course.
- *High-level synthesis*: When digital hardware blocks do need to be designed for an SoC, the traditional design methodology has been to design them at RTL level and then use logic synthesis, verification, and normal digital implementation flows to integrate them into SoCs. High-level synthesis has gone through interesting historical evolutions since the late 1980s at least [33], including considerable interest in the mid-1990s using hardware description languages as the input mechanism, a collapse, and growth in the more recent decade using C/C++/SystemC as inputs. It is now used as a serious part of the design flow for various complex hardware blocks, especially those in the dataplane

(e.g., various kinds of signal processing blocks). The chapter in the handbook on high-level synthesis is a good reference to the current state of the art here.

- *Verification growth and change*: Of course, as the SoC design grows in complexity, the verification challenge grows superlinearly. Many methods, some quite old, have seen wider adoption in the last decade. These include formal verification (both equivalence checking and property checking), use of hardware verification engines and emulators, hardware–software coverification, and virtual platforms. In addition, regular RTL and gate-level simulation has exploded for large SoCs. Many chapters in the EDA Handbook touch on these topics.
- *Back-end design flow*: Several important changes have occurred in back-end design tools and flow. In particular, advances in high-performance place and route, gate sizing, power-grid optimization, and clock gating and multiobjective optimizations are of major impact. For the most important optimizations, industry released large modern benchmark suites and organized multimonth research contests for graduate students. As a result, we now have much better understanding as to which methods work best and why. For participating teams, contests encourage sharper focus on empirical performance and allow researchers to identify simple yet impactful additions to mainstream techniques that would have been difficult to pursue otherwise. In most cases, the technique winning at the contest has been adopted in the industry within 2–3 years and often developed further by other academic researchers. Multiobjective optimization as a domain requires a keen understanding of available components and their empirical performance, as well as rigorous experimentation. It holds the potential for significant improvement in the back-end flow.
- *More Moore and more than Moore*: Despite the imminent predictions of its demise, Moore's law (or observation) [34] continues apace with each new technology generation. The time required for the number of gates on ICs to double may have stretched beyond the 1 year in the original observation, to 2 years in the 1970s and 3 years in the 2000s. Overall, process speed and thus the maximum MHz achievable with each process generation may have slowed down or stopped (or be limited by power concerns). SoCs are more complex in achievable gate count at 28 nm than at 40 nm, and a greater number of gates are possible at 16 nm. In addition, we have seen new process technologies—such as 3D FinFETs—emerge, which begin to fulfill the "more than Moore" predictions. Whether carbon nanotubes and transistors and gates built on them will become the workhorse technology at 7 or 5 nm or beyond is not clear, but something more than Moore is certainly emerging. These changes clearly influence the design process.

Although many changes in design trends have an impact on the design process, it retains many of the key characteristics and steps of a decade ago. The cumulative effect changes the emphasis on particular design disciplines and moves the dividing lines between them. Fundamentally, it has not changed the nature of the IC design process. This is likely to remain so in the future.

REFERENCES

1. M. Smith, *Application Specific Integrated Circuits*, Addison-Wesley, Reading, MA, 1997.
2. A. Kuehlmann, *The Best of ICCAD*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2003.
3. D. Thomas and P. Moorby, *The Verilog Hardware Description Language*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1996.
4. D. Pellerin and D. Taylor, *VHDL Made Easy*, Pearson Education, Upper Saddle River, NJ, 1996.
5. S. Sutherland, S. Davidson, and P. Flake, *SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2004.
6. T. Groetker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2002.
7. G. Peterson, P. Ashenden, and D. Teegarden, *The System Designer's Guide to VHDL-AMS*, Morgan Kaufman Publishers, San Francisco, CA, 2002.
8. K. Kundert and O. Zinke, *The Designer's Guide to Verilog-AMS*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2004.

9. M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1998.
10. J. Bergeron, *Writing Testbenches*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2003.
11. E. Clarke, O. Grumberg, and D. Peled, *Model Checking*, The MIT Press, Cambridge, MA, 1999.
12. S. Huang and K. Cheng, *Formal Equivalence Checking and Design Debugging*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1998.
13. R. Baker, H. Li, and D. Boyce, *CMOS Circuit Design, Layout, and Simulation*, Series on Microelectronic Systems, IEEE Press, New York, 1998.
14. L. Pillage, R. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*, McGraw-Hill, New York, 1995.
15. J. Elliott, *Understanding Behavioral Synthesis: A Practical Guide to High-Level Design*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000.
16. S. Devadas, A. Ghosh, and K. Keutzer, *Logic Synthesis*, McGraw-Hill, New York, 1994.
17. G. DeMicheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, 1994.
18. I. Sutherland, R. Sproull, and D. Harris, *Logical Effort: Defining Fast CMOS Circuits*, Academic Press, New York, 1999.
19. N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1999.
20. F. Nekoogar, *Timing Verification of Application-Specific Integrated Circuits (ASICs)*, Prentice-Hall PTR, Englewood Cliffs, NJ, 1999.
21. K. Roy and S. Prasad, *Low Power CMOS VLSI: Circuit Design*, Wiley, New York, 2000.
22. C.-K.Cheng, J. Lillis, S. Lin, and N. Chang, *Interconnect Analysis and Synthesis*, Wiley, New York, 2000.
23. W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, U.K., 1998.
24. M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*, Wiley, New York, 1995.
25. A. Wong, *Resolution Enhancement Techniques in Optical Lithography*, SPIE Press, Bellingham, WA, 2001.
26. International Technology Roadmap for Semiconductors (ITRS), 2004, http://public.itrs.net/.
27. International Technology Roadmap for Semiconductors (ITRS), 2012 update, http://www.itrs.net/Links/2012ITRS/Home2012.htm.
28. D. Nenni and P. McLennan, *Fabless: The Transformation of the Semiconductor Industry*, semiwiki.com, 2014.
29. A good source of information on EDA mergers and acquisitions, https://www.semiwiki.com/forum/showwiki.php?title=Semi+Wiki:EDA+Mergers+and+Acquisitions+Wiki.
30. H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, *Surviving the SOC Revolution: A Guide to Platform-Based Design*, Kluwer Academic Publishers (now Springer), Dordrecht, the Netherlands, 1999.
31. I. Bolsens, FPGA, a future proof programmable system fabric, Talk given at Georgia Tech, Atlanta, GA, March 2005, slides available at: http://users.ece.gatech.edu/limsk/crest/talks/georgiafinal.pdf. (Accessed on October, 2014).
32. I. Bolsens, The all programmable SoC—At the heart of next generation embedded systems, *Electronic Design Process Symposium*, Monterey, CA, April 2013, slides available at: http://www.eda.org/edps/edp2013/Papers/Keynote%200%20FINAL%20for%20Ivo%20Bolsens.pdf. (Accessed on October, 2014).
33. G. Martin and G. Smith, High-level synthesis: Past, present, and future, *IEEE Design and Test*, 26(4), 18–25, July 2009.
34. A good writeup is at http://en.wikipedia.org/wiki/Moore%27s_law, including a reference to the original article.