Computer-Aided Design for VLSI

Kartik Mohanram
Dept. of Electrical and Computer Engineering
Rice University

Today

- VLSI design flow
  - System drivers [ITRS roadmap]
- EDA Tools
  - Synopsys, Cadence, Mentor
- Conferences
  - DAC, ICCAD, DATE, ITC, VTS, ISPD, ISLPED, …

System Drivers

- Future semiconductor manufacturing and design technology capability is developed in response to economic drivers within the worldwide semiconductor industry.
- 3 Product classes previously
  - Microprocessor (MPU)
  - Dynamic random-access memory (DRAM)
  - Application-specific integrated circuit (ASIC)
- Some mention of system-on-chip (SoC) and analog/mixed-signal (AMS) circuits
- Assumption – technological advances need only be straight ahead and linear
  - Deployed in all semiconductor products
  - Specifics of product classes not required

System Drivers

- Today, introduction of new technology solutions is increasingly application-driven, with products for different markets using different combinations of technologies at different times
  - General-purpose digital microprocessors for personal computers are being joined as drivers by mixed-signal systems for wireless communication and embedded applications
  - Low-power battery-powered mobile devices
  - In-house, single-source chip designs are being supplanted by SoC designs that incorporate building blocks from multiple sources

System Drivers [ITRS 2001]

- 3 system drivers
  - High-volume custom microprocessor (MPU)
  - Analog/mixed-signal (AMS)
  - System-on-chip (SoC)
- What about
  - Application-specific integrated circuits (ASICs)?
  - High-volume custom memory (DRAM)?
    - Commodity nature

Driver Classes

- Driver classes distinguished according to
  - Cost = manufacturing cost + design cost
  - Manufacturing cost
    - Non-recurring engineering (NRE) cost (masks, tools, etc.) +
    - Silicon cost (raw wafers + processing + test)
  - System cost = F(functionality, No. I/Os, packaging cost, power, speed, etc.)
  - Time-to-market
  - Production volume

MPUs

- Performance and manufacturing cost issues outweigh design or other NRE cost issues
  - Primarily because these chips can potentially produce large profits
    - From very large sales volumes
  - Large volumes alone neither necessary nor sufficient to warrant custom design, special process engineering and equipment, etc.
    - Key – expected return on combined NRE and manufacturing investment must be positive
- 3 dominant sub-classes
  - MPUs, memory, and re-programmable devices (e.g., field-programmable gate arrays (FPGA))

MPUs

- Use THE most aggressive design styles and manufacturing technologies
  - It is for these high-volume parts that
    - Changes to the manufacturing flow are made
    - New design styles and supporting tools are created (the large revenue streams can pay for new tool creation)
    - Subtle circuits' issues are uncovered (not all risks taken by designers work out).
  - Thus, while MPUs (and high-volume custom designs in general) are extremely labor-intensive, they create new technology and automation methods (in both design and fabrication) that are leveraged by the entire industry.

Memory (DRAM)

- Special class of high-volume custom design
  - Very high replication rate of basic memory cells and supporting circuits
  - Repeated millions of time on a chip
  - Millions sold
  - Extraordinary custom design
    - Separate fabrication lines for DRAM devices
    - Most careful circuit engineering needed to ensure correct operation

Analog/mixed-signal (AMS) Circuits

- Different design and process technology demands than digital circuits
  - Technology scaling always desirable for digital circuits – reduced power, area, and delay
  - Not necessarily a plus for analog circuits
    - Dealing with precision requirements or signals from a fixed voltage range is more difficult with scaled voltage supplies.
    - Scaling AMS circuits into new technologies difficult
    - AMS circuits (e.g., RF and embedded passives) and process technologies (e.g., silicon-germanium) also present severe challenges to cost-effective CMOS integration..

Application-Specific Integrated Circuits

- Most digital designs in last decade – ASICs
- Connotes both a business model (with particular handoff from design team to ASIC foundry) and
- A design methodology where
  - The designer works predominantly at the functional level, with Verilog/VHDL or higher level hardware description languages and
  - Invokes automatic logic synthesis and place-and-route with a standard-cell methodology
  - For economic reasons, custom functions are rarely created
  - Reducing design cost and design risk is paramount

Application-Specific Integrated Circuits

- ASIC design is characterized by
  - Relatively conservative design methods and design goals (note differences in clock frequency and layout density between MPUs and ASICs)
  - Aggressive use of technology – moving to a scaled technology is a cheap way of achieving a better (smaller, lower power, and faster) part with little design risk (note convergence of MPU and ASIC process geometries)
  - Latter half of the 1990s, ASICs and SoCs converging in terms of content, process technology, and design methodology
  - ASIC/SoC and MPU design methodologies are also converging to a common hierarchical ASIC/SoC methodology
    - Customer-owned tooling business models (ASIC side)
    - Tool limitations faced by both methodologies

System-on-Chip (SoC) Design

- SoC is a yet-evolving product class and design style
  - Integrates pieces of technology from other system driver classes (e.g., MPU, memory, AMS, and re-programmable) into a wide range of high-complexity, high-value semiconductor products
- Manufacturing and design technologies for SoC are typically developed originally for high-volume custom drivers
- The SoC driver class most closely resembles, and is evolved most directly from, the ASIC category since
  - Reduced design costs and
  - Higher levels of system integration are its principal goals.

System-on-Chip Design

- Primary differences from ASIC
  - Goal of the SoC paradigm is to maximize reuse of existing blocks or cores, i.e., to minimize the amount of a design that is newly or directly created
  - Reused blocks in include analog and high-volume custom cores, as well as blocks of software technology
  - A key challenge is to design, create and maintain reusable blocks or cores so that they are available to SOC designers
  - The utility of SoC also depends on validation for reuse-based SoC designs being easier than for equivalent from-scratch designs

System-on-Chip Design

- SOC's a confluence of previous product classes in several ways
  - SoCs integrate building blocks from the other system driver classes, subsuming the ASIC category
  - The quality gap between full-custom and ASIC/SoC is diminishing:
    - [ITRS 01] models overall ASIC/SoC and MPU logic densities as being equal
  - Custom quality on an ASIC schedule achieved by on-the-fly (liquid) standard-cell methodologies
  - MPUs evolving into SOCs
    - MPUs designed as cores to be included in SOCs
    - MPUs themselves designed as SOCs to improve reuse and design productivity
    - Alpha 21364, dual- and quad-core Intel/AMD chips

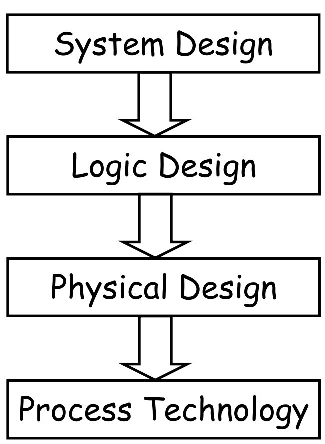
SoC System Driver

- SoC driver class characterized by
  - Heavy reuse of intellectual property (IP) to improve design productivity
  - System integration that potentially encompasses heterogeneous technologies
  - SoCs exist to provide low cost and high integration
    - Cost considerations drive deployment of low-power processes and low-cost packaging solutions
    - Fast turnaround time design methodologies

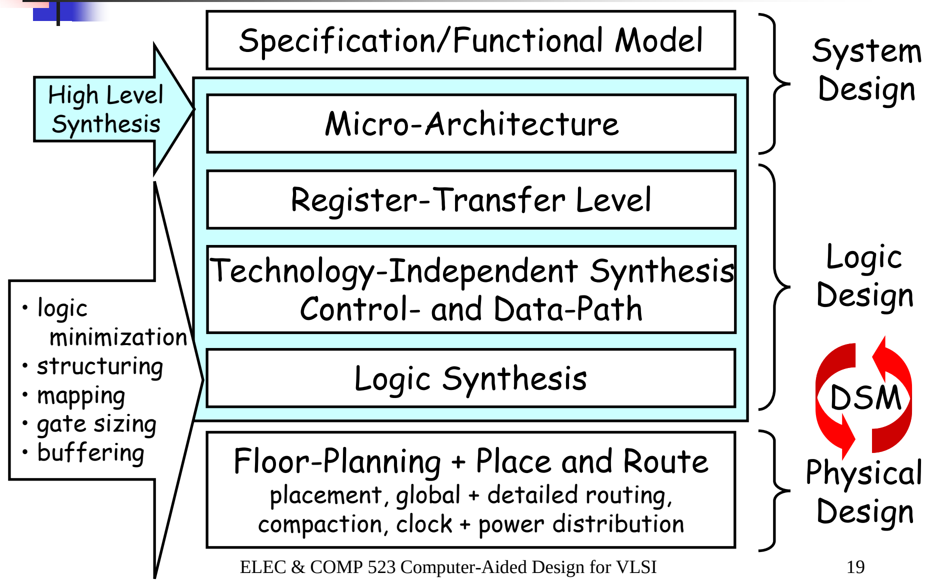
SoC System Driver

- SoCs necessitate new standards and methodologies for IP description, IP test (including built-in self-test and self-repair), block interface synthesis, etc.
- Integration considerations drive the demand for heterogeneous technologies (flash, DRAM, MEMS, ferroelectric RAM (FRAM), chemical sensors, etc.) in which particular system components (memory, sensors, etc.) are implemented, as well as the need for chip-package co-optimization
- In summary, SoC is the driver for convergence of multiple technologies not only in the same system package, but potentially in the same manufacturing process
  - Three variants driven by multi-technology integration (MT), high performance (HP), and low-power, low-cost (LP-LC) (by no means disjoint)

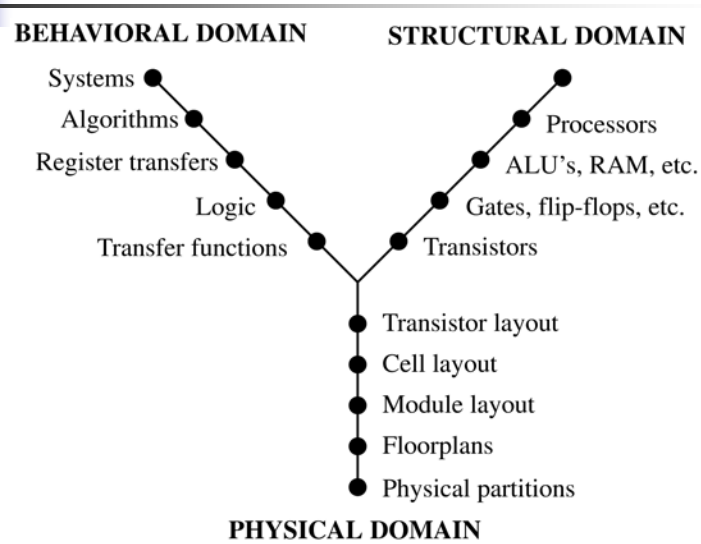
Design Flow



Traditional ASIC Design Flow



Y-chart



Electronic Design Automation (vs. CAD!)

- Enabled and propelled growth of the electronics industry
- $3 billion EDA market
  - Synopsys + Avant! (~30%)
  - Cadence (~30%)
  - Mentor Graphics (~25%)
- Holy grail – an automated design flow connecting the front end, where Synopsys was historically the strongest, and the back end, where Cadence was historically the strongest

Synthesis, Place, and Route

- Over the last three years, many of the big vendors in electronic design automation have focused their R&D efforts on all-in-one RTL-to-GDSII tool suites that tie together logical or RTL design and synthesis with floor-planning, placement, and routing.
- The idea is that linking these tools tightly in a single flow will better equip engineers to deal with timing, power, and signal-integrity issues of very deep-submicron (VDSM) fab processes by letting them write RTL and perform synthesis, placement, and routing in a single pass.

What is GDSII?

- GDSII is a binary file format
  - Data interchange format used to transfer mask-design data between IC designer and fab
- At the fab, GDSII data converted to machine-readable language called CATS (Computer Aided Transcription Software)
  - CATS transcribes data for photomask systems used for manufacturing semiconductors .
- GDSII stands for Graphic Design System II
  - Alternative names are "Calma Stream Format" (CSF) or just "Stream Format"

Synopsys' Perspective (or Pitch)

- The growing complexity of today's SoC designs often requires validating the system functionality and architecture before committing to a detailed micro-architecture.
- System-level design – create an executable specification, then convert to RTL with complete functional verification
- RTL design – optimize and validate the design's micro-architecture, evaluate RTL quality, and deploy intelligent functional verification
- Physical design – carry through the design from RTL to GDSII.

Synopsys' System-Level Design

- Design quality and schedule
  - Specification closure at block- and chip-levels
  - Minimize iterations between architecture and RTL implementation
- Developing advanced algorithms
- Functional modeling and architectural exploration
  - Evaluating and quantifying architectural performance
  - Micro-architecture optimization
  - Developing executable specifications w/migration to RTL
- Partitioning and co-designing hardware and software

Synopsys' System-Level Design

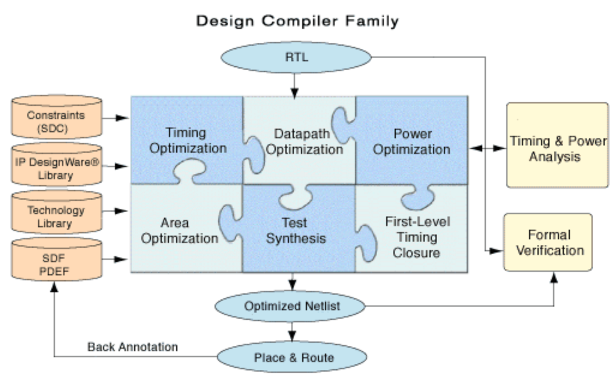
- Developing floating and fixed-point design representations
- Accelerating software development with FPGA prototyping
- Evaluating and integrating soft IP components
  - Predictability for SoC integration early in the design cycle
- Developing and optimizing new RTL code
  - Qualifying and optimizing existing RTL blocks
- Developing intelligent test-benches
- Improving verification methodology and coverage

Synopsys' Synthesis Solution

Design Task	Design Step	Tools
Design Implementation through placed gates	Design Planning	JupiterXT™, Floorplan Compiler
	Synthesis	Design Compiler®, Module Compiler™
	Static Timing Analysis	PrimeTime®
	Formal Verification	Formality®
	Physical Synthesis	Physical Compiler®
Testability	DFT analysis, ATPG	DFT Compiler™, TetraMax™
	JTAG insertion	BSD Compiler
Power	Power Analysis/Optimization	PrimePower, Power Compiler™, AstroRail™
Physical Implementation & Physical Verification	Place and Route	Astro™, Apollo II™
	Clock Tree Generation	CTS, Clock Tree Compiler
	Top level routing	Astro™, Apollo II™, Columbia™
	Extraction	STAR RCXT™
	DRC/LVS	Hercules™
	Signal Integrity	Primetime-SI, AstroXtalk™, AstroRail™
	Layout editor	Astro™, Apollo II™, Columbia™
	IPO/ECO processing	Apollo II™, Astro™, Physical Compiler®

- Reading
  - http://www.synopsys.com/sps/pdf/physical_design_ds.pdf

Synopsys' Synthesis Solution



- Source
  - http://www.synopsys.com/products/logic/design_compiler.html
  - Also see Design Compiler backgrounder http://www.synopsys.com/products/logic/design_comp_tb.pdf

Synopsys' Physical Design Platform

- Netlist to GDSII
- Support for netlist, placed-gates, or GDSII hand-offs
- Qualifying libraries and existing RTL
- Capturing missing constraints
- Design planning (flat, hierarchical, and virtual flat)
- Planning and analyzing power distribution
- Physical Compiler-based design optimization
- Hardening of soft IP and IP integration
- Full-chip static-timing analysis (STA)
- Place and route
- Generating and optimizing clock trees
- Physical verification

Physical Verification

- Design rule checker (DRC)
- Electrical Rule Checking (ERC)
- Layout versus Schematic (LVS)
- Reading – Mentor's Calibre DRC and LVS

http://www.mentor.com/calibre/datasheets/calibre/Calibre_DRCLVS.pdf

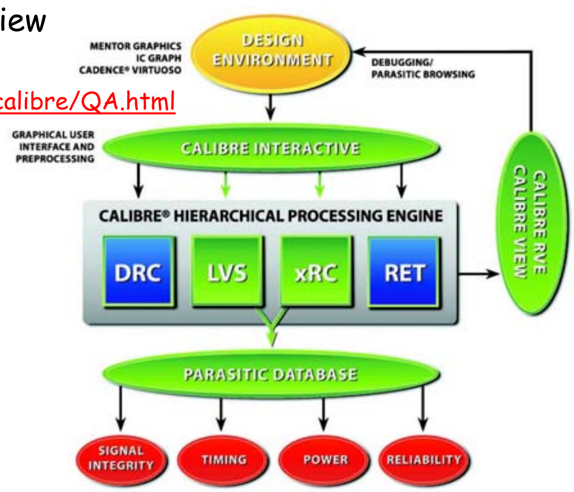
Physical Verification

- Backannotation
  - Allows the extraction of parasitics, etc. from interconnects in the layout and add them as properties to corresponding objects in the schematic
  - Facilitates more accurate simulation
  - Can choose to optimize (design as well as layout) further to improve chip performance
- Reading – Mentor's Calibre xRC, xCalibre
  - Full-chip parasitic extraction
  - Reading
    http://www.mentor.com/xcalibre/Calibre_xRC_ds.pdf
    http://www.mentor.com/xcalibre/xcalibre_ds.pdf

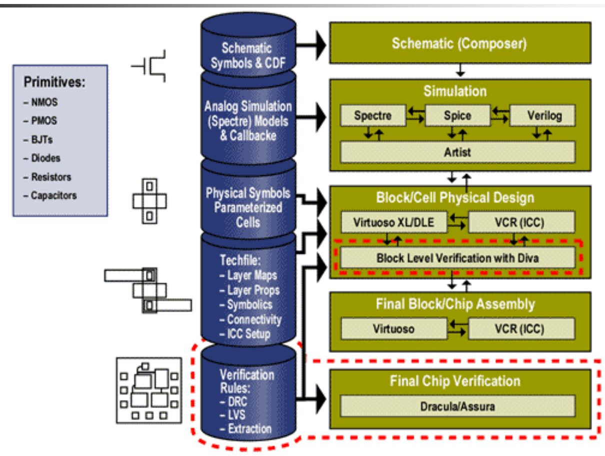
Physical Verification

- Calibre xRC overview
  - FAQs

http://www.mentor.com/xcalibre/QA.html

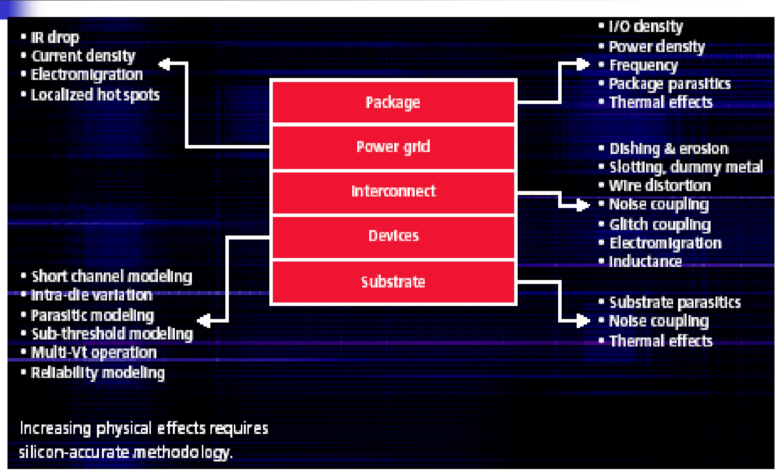


Physical Verification



- Cadence – physical verification deck
  http://www.cadence.com/infosheets/is_phys_ver_srv.html

Cadence's Virtuoso Platform



http://www.cadence.com/brochure/4869_VirtuosoBroch_FNL.pdf

Design Synthesis and Silicon Compilation

- Two schools of thought
  - Design process difficult to automate
  - Human designer main source of design knowledge
  - Increase productivity with sophisticated tools for
    - Design capture, verification, analysis, and optimization
    - Logic simulators, design-rule checkers (DRCs), compactors, etc.
- Gajski's paper
  - Automatic generation of VLSI designs
    - symbolic layout, schematic, behavioral description, instruction set, architecture

Advantages

- Usability (from a tool perspective)
  - Designers work at higher levels of abstraction
  - Tools then take charge
- Design quality
  - Correct-by-construction (tall order!)
  - Design errors easier to correct at higher levels
- Designer productivity
  - Time-to-market

Disadvantages

- Design space exploration
- Different compilers for different process technologies
- Silicon compilers hard to integrate into present day design-flow
  - Transition from schematic-/layout-capture to silicon compilation

Where does the SoC Methodology Fit In?

- SoC captures this to some extent
- Hard vs. soft IP cores
- IP blocks can be delivered in one of two main types
  - Hard IP, where the final GDSII layout is given to the customer
  - Soft IP, where the RTL source code is released to the customer who is then responsible for hardening the IP (synthesizing it using a suitable gate library and running place and route to obtain the final layout).
- Depending on the type of IP being used, the chip integrator has to deal quite differently with issues such as verification, timing analysis, and manufacturing test.