DAT110
Methods for Electronic System Design and Verification

Per Larsson-Edefors
VLSI Research Group

Lecture 1: Course Introduction. Electronic System Design.

Why the Methods course?

Motivation

- Billions of transistors on single chips.
  - First reaction: “Great! More performance and functionality!”
  - ... but how do we make use of such a complex platform?
- Electronic system designers are forced to make use of design automation tools to manage design complexity and meet, for example, strict timing, power and time-to-market budgets.
- To apply the right tools in the right context and in the right sequence has become a methodological challenge that rivals traditional design challenges intrinsic in logic and circuit design.

In This Course, You will for Example...

- get to know the algorithmic principles of a number of important EDA concepts, such as behavioral and logic synthesis, logic simulation, static timing analysis, timing closure and power analysis.
- learn about contemporary EDA flows and their fundamental weaknesses and strengths.
- learn how to apply appropriate EDA tools to design and verification problems.
- ... three more goals, see the portal.

Strong Links to EESD Goals ...

- "... proficient in the use of various state-of-the-art Electronic Design Automation (EDA) tools used in industry"
- "... aware of the fundamental limitations of ... the design tools and methodologies... that represent current best practice"
- "... able to carry out qualified industrial tasks within given constraints by applying suitable methods..."
- "... clearly and unambiguously communicate their conclusions, and the knowledge and rationale underpinning these conclusions"
- "...able to work in teams with different engineering competencies for solving complex engineering tasks..."

Terminology Examples

- EDA = Electronic Design Automation.
- CAD = Computer Aided Design.
- RTL = Register Transfer Level.
- P&R = Place and Route.
- HLS = High-Level Synthesis.
- Design flow = EDA tools arranged in sequence using scripts.
- Synthesis = refinement (often a reference to automated refinement).
- Verification = check that requirements are fulfilled.
- Validation = check that intent is respected.
- Heuristic = a problem-solving algorithm with unpredictable outcome.

Pedagogics and examination

Teacher Team

- Instructor.
  - Per Larsson-Edefors.
  - Consultation hours: Mondays & Tuesdays 1:15-2:00pm (any exceptions will be listed on PP).
- Teaching assistants (labs).
  - Christoffer Fougstedt.
  - Erik Börjeson.
  - Consultation hours: Thursday Nov. 8, 15, and 22, 2:00-3:00pm.
- Technical writing lecturer (term papers).
  - Anne Hsu Nilsson (Gothenburg University).

Course Organization

- Lectures.
  - Design and verification context of advanced electronic systems.
- Lab exercises.
  - Comprehensive hands-on training: RTL to P&R.
  - Best design practice using state-of-the-art EDA tools.
  - Emphasis on properties like timing and power/energy.
- Term paper.
  - Study in optional technical area.
  - Training in technical writing and oral presentation.
  - *More info on this on Lecture 3 next week.*

Two-Part Examination

- Lab exercises on design flow, including RTL design, synthesis and place and route for ASICs.
  60% of grade is based on quality of preparation, execution and outcome in terms of lab report.
- Group work (3 students) on selected topics, including the writing of a term paper and an oral presentation.
  40% of grade is based on quality of term paper and presentation.

No final written exam!

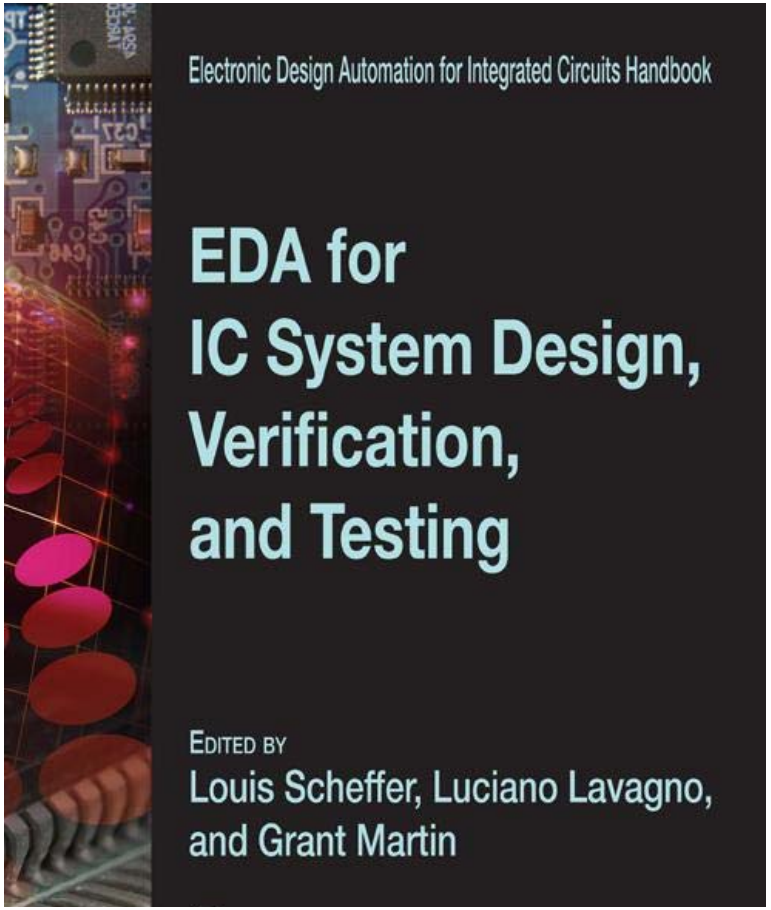
Work continuously for optimal learning!

Main "Textbook"

EDA for IC System Design, Verification, and Testing (Volume 1)

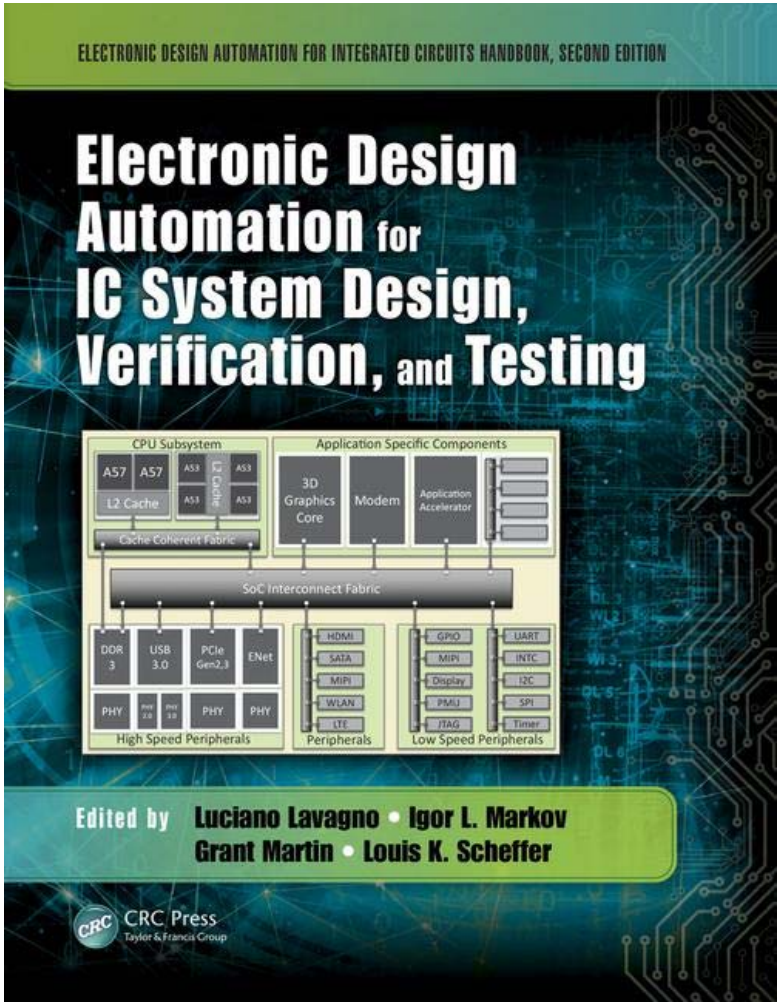
+

EDA for IC Implementation, Circuit Design, and Process Technology (Volume 2)



*(Downloadable as e-book within Chalmers; see CHANS: **chans.lib.chalmers.se**)*

New 2016 Edition (@ 300 USD !)

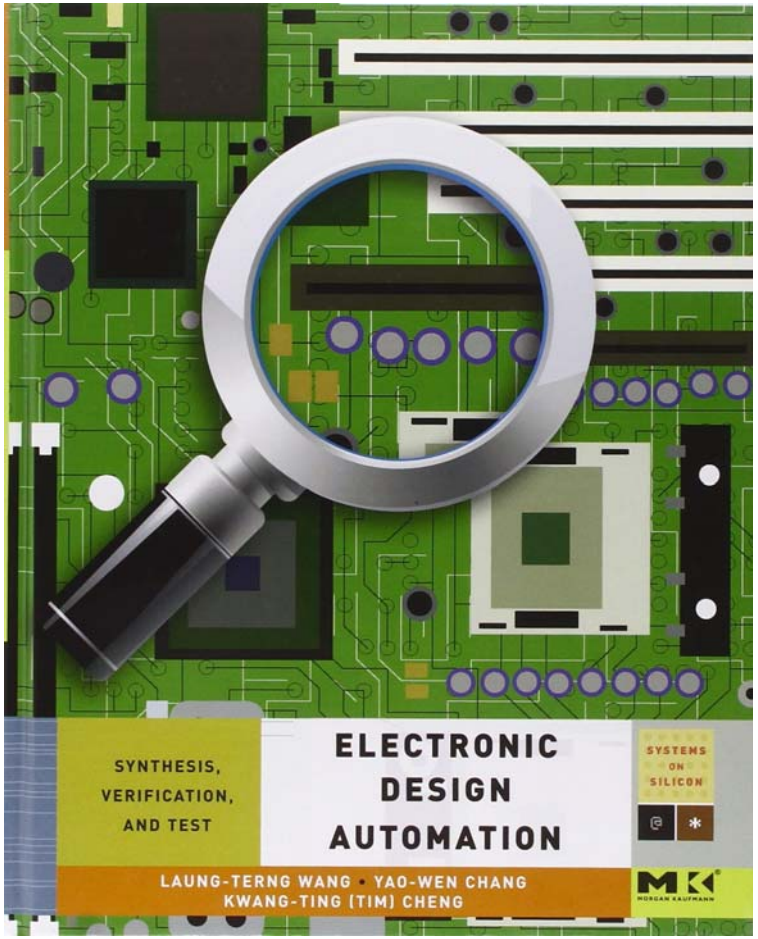


Volume 1 available as e-book.

Go to CHANS and search for

Electronic design automation for
IC system design, verification, and testing

CHAPTERS FROM AN "EXTRA BOOK"



(Also downloadable as e-book within Chalmers;)

ASIC design flow

To Deal with Billions of MOSFETs

- Consider abstractions.
  - Signals are digital.
  - MOSFET transistors are switches.
  - Gates are characterized as stand-alone elements (cell library).
  - The overall behavior is synchronous.
  - Hardware description languages (VHDL, Verilog).
- Consider other methods.
  - Partition the design (“divide-and-conquer”) - use a design hierarchy.
  - Use constraints.

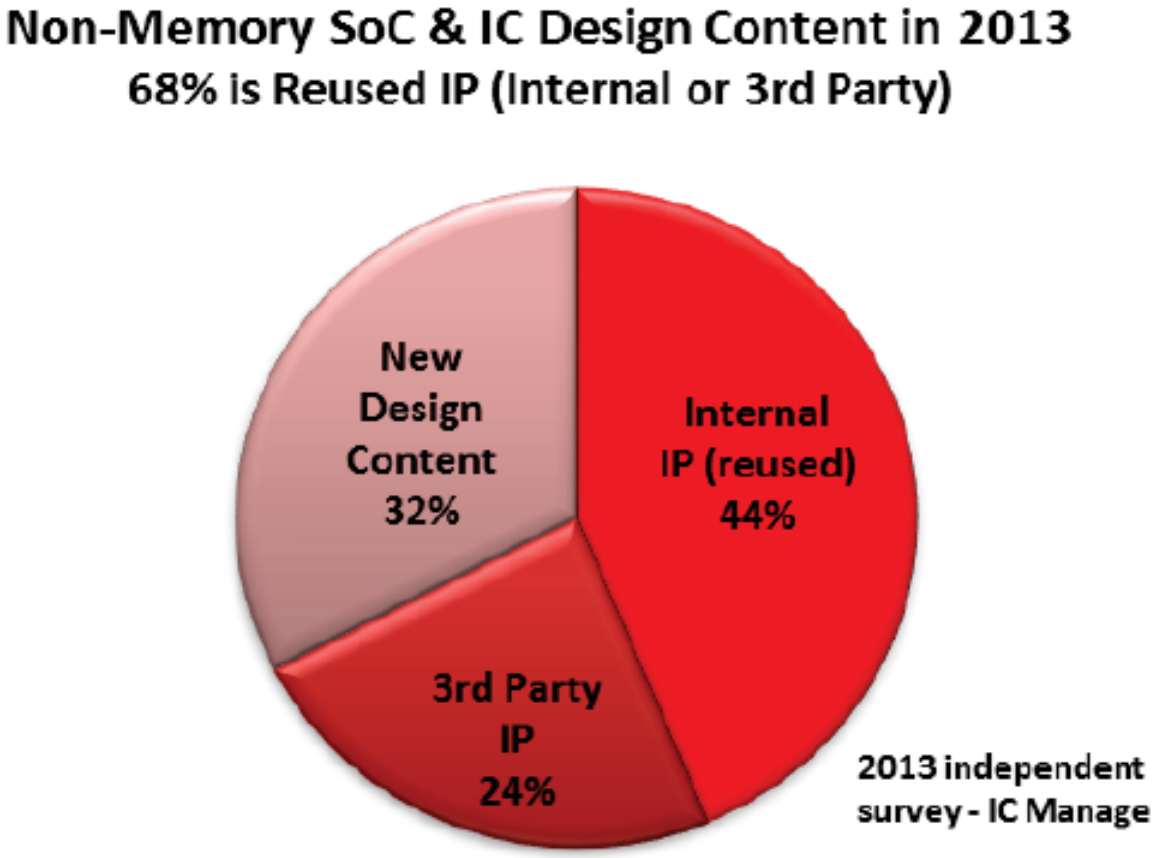
Design Hierarchy …

- helps handle complexity, hides details and reduces the number of things to handle at any given time.
  - enables top-down design.
  - enables bottom-up design.
- encourages use of modules. Use of “black boxes” makes implementation and verification easier.
- *Preparatory assignment:*
  Prepare system block diagram;
  consider overall function and timing.

Design Approaches

- Bottom up.
  - for example, standard-cell circuits are first laid out and then introduced at more abstract levels of design.
  - slow approach if entirely sequential,...
  - but important for new implementations.
- Top down.
  - for example, bring an algorithm to silicon.
  - identify and bring in intellectual property (IP) blocks.
  - what technology assumptions are made in early stages of design?
- Middle out; a combination of bottom-up and top-down.

Importance of Re-Use and IP Blocks



- Merging EDA and semiconductor IP strategically important: Synopsys made $500M on IP products in 2016.

Electronic Design Automation

- Electronic systems are complex and computer-aided methods are necessary: **Electronic Design Automation**.
- Levels of design and verification.
  - Electronic system level (ESL).
  - Algorithmic level - dataflow oriented.
  - Behavioral level - a notion of sequence exists.
  - Transaction level - captures interface issues, untimed to timed.
  - Architectural level - hardware structures are visible.
  - Register transfer level (RTL).
  - Gate level.
  - Circuit level.
  - Layout level.

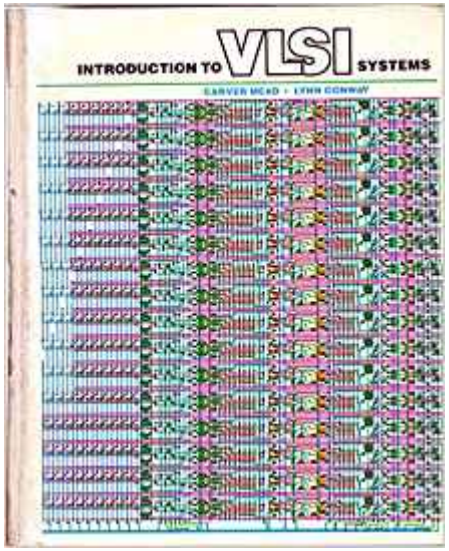
Development Phases

Design
is easy
-
minor part of
development
time
“30%”

Verification
is difficult
-
major part of
development
time
“70%”

The First EDA Vendors

- Work by Mead and Conway in the end of 70s. Seminal book “Introduction to VLSI Systems” from 1979: Key concepts like MPW, silicon compilation and lambda-based design.



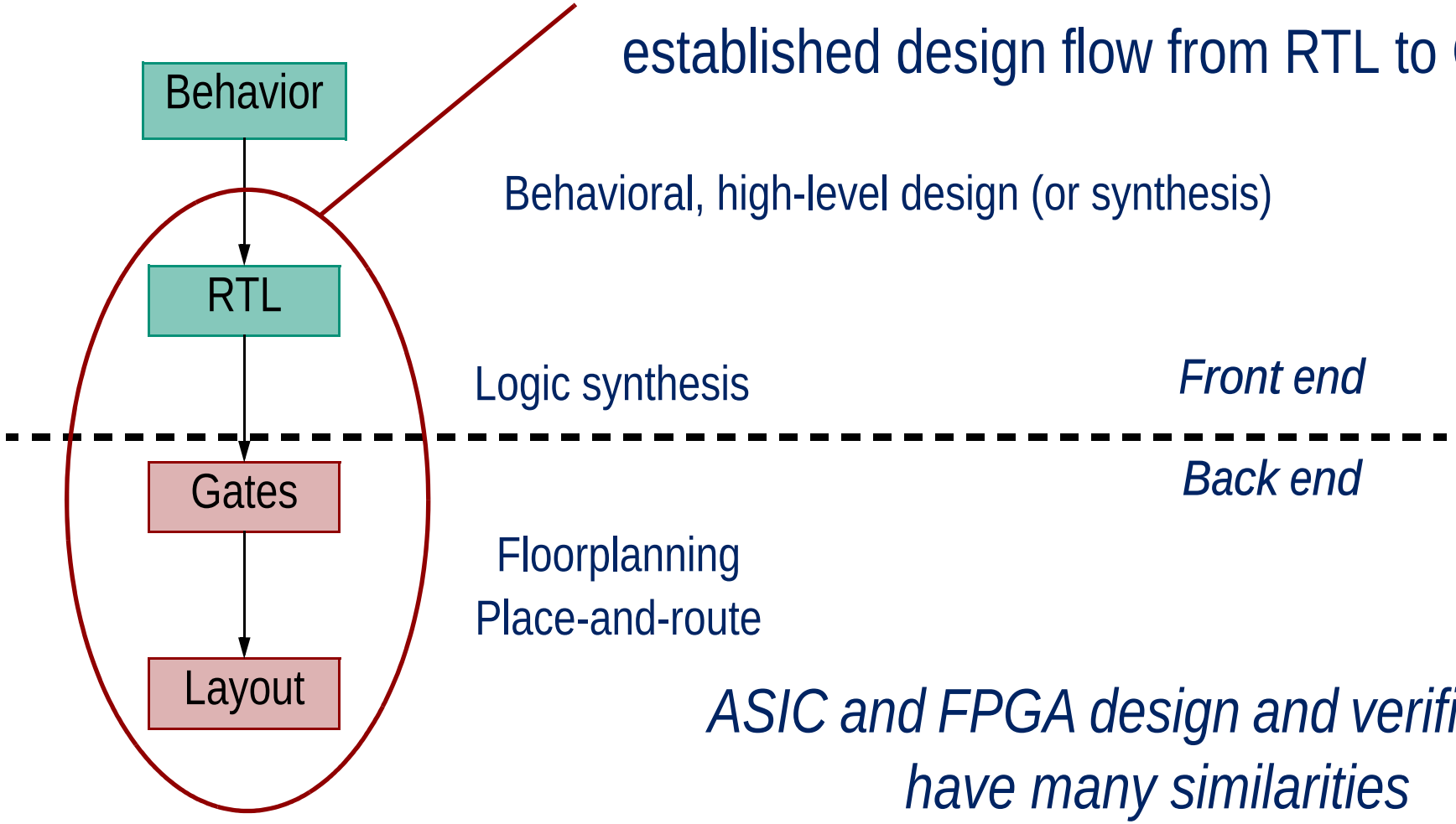
- EDA businesses spun out of major electronics corporations:
  - Daisy Systems and Valid Logic Systems (schematic capture, logic simulation) with proprietary HW/SW systems.
  - Mentor Graphics: Spin off (1981) from Tektronix (Beaverton OR). IDEA 1000 simulation software running on Apollo workstations.
  - VLSI Technology Inc.: Early foundry. Cell library design (IP + EDA).

Younger EDA Vendors

- Early EDA businesses had problems caused by poor scalability; dedicated software (assembly) running on rigid hardware.
  Two successful companies offered software only:
  - EDAC (Dracula layout checker) 1982 and SDC (layout design framework) merged in 1988 and became Cadence.
    Some key acquisitions: Gateway (with Verilog) in 1989.
    Valid Logic Systems in 1991.
  - Optimal Solutions, a spin off from General Electric in NC (1986).
    After moving to Mountain View CA 1987, this became Synopsys.
    The SOCRATES system of GE was a technology mapper [Lecture 4] and became the starting point of Synopsys Design Compiler.
    Some key acquisitions: Epic (1997), Avanti (2002), Magma (2012).

Front-end and Back-end ASIC Design

Our exercises are focused on the established design flow from RTL to GDSII



ASIC and FPGA design and verification have many similarities

Hardware Description Languages

- HDLs can describe hardware from behavioral to gate level, but mainly the range is RTL to gate level. There are two dominant HDLs:
  - VHDL (VHSIC HDL): specification/documentation, based on Ada, inception 1981, US defence contract involving e.g. IBM and TI, became IEEE standard in 1987.
  - Verilog HDL: for simulation, inspired by C, developed at Gateway Design Automation 1983-85, Cadence acquires Gateway in 1989 and makes Verilog HDL public in May 1990. Cadence launches the Leapfrog VHDL simulator in 1993.

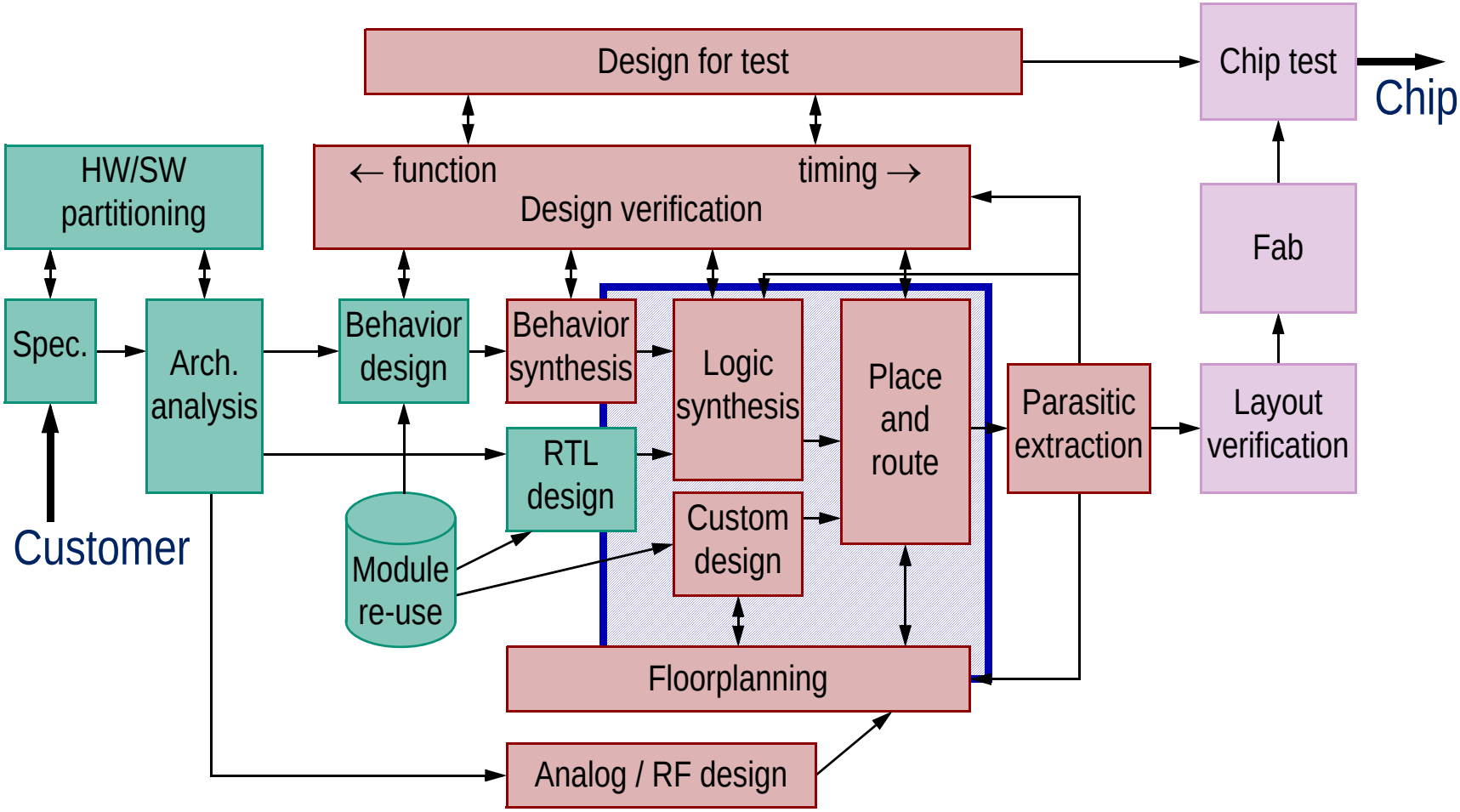
HARDWARE DESCRIPTION LANGUAGE

- You have hardware structure and/or functionality in mind and need to describe and verify your concept.
- Description != programming.
- Behavior (software) vs structural (RTL) code.
- Essential advice:
  Always picture your system, as block diagram,
  before writing RTL code (central to *Preparatory assignment*).

Synthesis - "Compilation of HW"

- General design flow.
  1. Behavior (C code or HDL) $\rightarrow$ RTL — High-level synthesis
  2. RTL $\rightarrow$ Physical implementation — Logic synthesis
- Conventional design flow for ASICs.
  1. RTL-specification $\rightarrow$ Generic gate netlist.
  2. Generic gate netlist $\rightarrow$ Cell library.
  3. Cell library $\rightarrow$ Placement & routing.

Conventional Logic/Gate Design



Lab exercises on ASIC design flow

Exercises

- Preparation (Study Week 1)
  - ALU implementation and timing plan.
- Exercise 1: ALU Design and Verification (SW 2-3)
  - HDL descriptions/testbenches and functional verification.
- Exercise 2: Synthesis and Timing Analysis (SW 4)
  - Logic synthesis with timing constraints, STA and timing simulation.
- Exercise 3: Timing Closure and Power Analysis (SW 5)
  - Timing closure and power dissipation analysis.
- Exercise 4: ALU Place and Route (SW 6)

Computer Lab Schedule

- Supervised lab hours:
  - Exercise 1a: Wednesday 13-17 and Friday 13-17 in SW 2.
  - Exercise 1b: Wednesday 13-17 and Friday 13-17 in SW 3.
  - Exercise 2: Wednesday 13-17 and Friday 13-17 in SW 4.
  - Exercise 3: Wednesday 13-17 and Friday 13-17 in SW 5.
  - Exercise 4: Wednesday 13-17 and Friday 13-17 in SW 6.
  - *Backup: Wednesday 13-17 and Friday 13-17 in SW 7.*
- Lab hall: 4220.

Lab Exercises - Groups

- Lab groups will be formed by the teacher team.
  - List of groups published in the end of SW1.
  - Half the class on Wednesdays; other half on Fridays.
  - Let me know, via email, if you have conflicts any of these days.
  - If you did not attend MCC092, you will not have EDA access: Meeting with Lars S on Friday 9:30am.

Lab Examination - Hand-Ins

- Preparatory assignment: Individual solution submitted in PingPong no later than Monday Nov. 12 (SW2).
  - Use LaTeX and drawing tool; generate PDF file.
  - No resubmission.
  - Short individual feedback (in PingPong) before Wednesday lab SW2.
- VHDL hand-in: Individual submission of VHDL files in PingPong for the ALU and the testbench, no later than Friday Nov. 23 (SW3).
  - No resubmission.
  - Individual Go/No go (in PingPong) before Wednesday lab SW4.

Lab Examination - PingPong Log Book

! Log book in PingPong: Four individual log book entries.

- Consider your lab work in the context of
  lab learning outcomes (see lab memo)
  and supporting lectures:
  Suggested content: Summary of lab work. Key results.
  Reflections on what you have learnt.
- Short entries (< 300 words). Only text, no figures.
  Plain ASCII text, no attachments.
- Write entry no later than one week after you complete lab $\Rightarrow$
  I will give feedback during that week.
- Do not respond to my feedback, but consider this for your lab report.

Lab Examination - Lab Report

! Lab report:

Submit individually crafted report no later than Friday Dec. 21.

- Describe your lab work and its result in the context of lab learning outcomes and supporting lectures.
  Use book chapters as references for your discussions.
- The report should treat all four lab exercises holistically.
- Output format should be PDF. Use, e.g., LaTeX.
- A strict limit is 4 pages.
- No resubmission.
- Limited feedback (continuous feedback on the log book entries).

Grading Principles for Lab Report

3. Report should convey understanding of work performed.
   - Relate your lab work to, at least,
     Vol. 1/Ch. 2 of main textbook (2016).
4. Report should convey good understanding of work performed.
   - Relate your lab work to, at least,
     Vol. 1/Ch. 2 (2016) and Vol. 2/Ch. 1 (2006).
5. Report should convey very good understanding of work performed.
   - Relate your lab work to, at least,
     Vol. 1/Ch. 2 (2016), Vol. 2/Ch. 1 (2006) and Vol. 2/Ch. 10 (2006).

Lab Grading Function

- Grading on lab part:
  - 20% preparatory assignment.
  - 30% VHDL hand-in.
  - 50% lab report and log book.

- *The term paper work will be treated on Lecture 3 next week.*

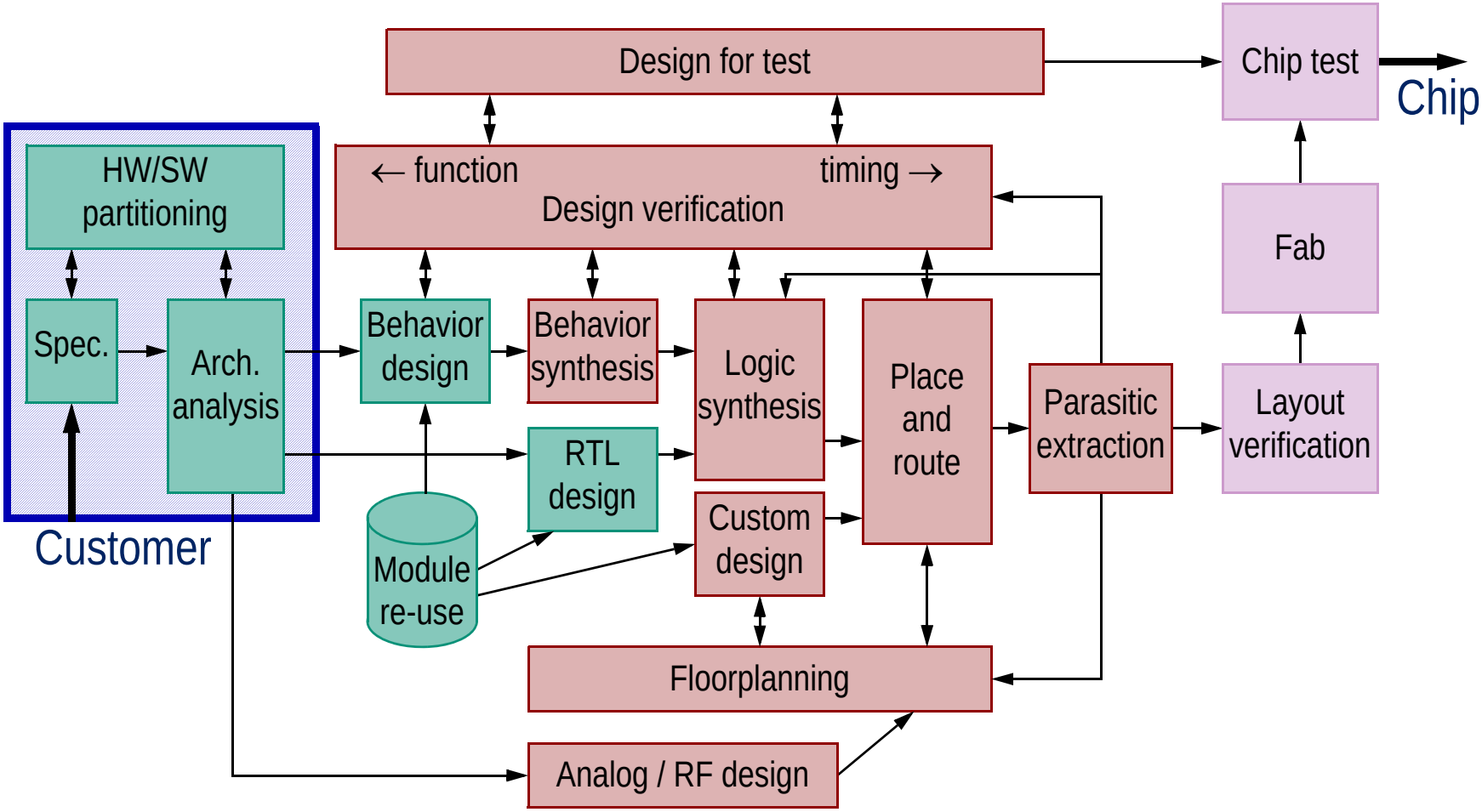
Lecture context

(Tentative) Lecture Plan

SW1	Today's lecture
	Functional verification
SW2	Term paper information. Ethics assignment. Practical advice for lab exercises.
	Synthesis
SW3	Timing
	Power and energy. Variability.
SW4	Physical design
	Reliability and test
SW5	Technical writing with Anne Hsu Nilsson
	Discrete mathematics and optimization strategies for EDA
SW6	*Term paper presentations* * 2 (starting time may change)
SW7	Term paper writing workshop with Anne Hsu Nilsson
	Term paper presentations (starting time may change)

Electronic system design

System Design



Electronic Systems

- Data in focus (function-based design):
  High-throughput dataflow-oriented (DSP) systems
  which are homogeneous and regular in organization.
- Control and data (architecture-based design):
  Processor-based systems
  for which software is essential and
  a host of heterogeneous blocks are needed.
  - More and more DSP systems take this approach.

Function-Based DSP-Oriented Design

1. Describe / analyze floating-point systems using MATLAB.
2. Develop fixed-point HW models.
3. Develop VHDL/Verilog blocks.

- EDA for DSP HW:
  - Algorithms only:
    Simulink, SPW (Comdisco 1988->Cadence->CoWare->Synopsys),
    System Studio (Synopsys).
  - HLS [Lecture 4] (more versatile: control logic + memory interfaces):
    Forte Cynthesizer (now in Stratus from Cadence),
    Catapult C (Mentor->Calypto->Mentor (Siemens)).

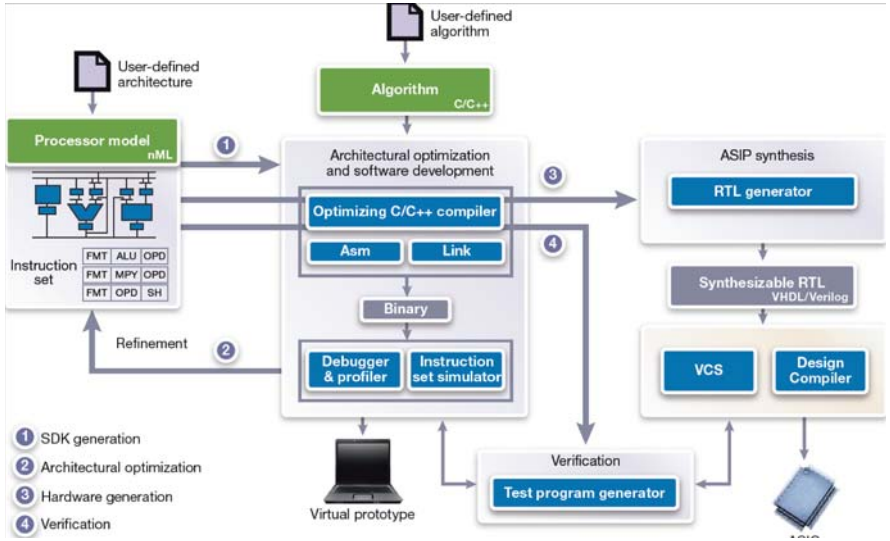
Architecture-Based Design

- Combination of processors and other blocks (accelerators, peripherals, etc.) on a System on Chip (SoC).
- Requires codesign of hardware and software (HW/SW).
- Sequential design process slow. How can we design SW without having access to final HW?
  - HW models allow SW to be implemented, executed and profiled.
  - Virtual prototypes: HW is emulated using virtual components.
  - System C: HW and SW can codesigned.
- Read more [Sec. 5.1 of Ch5_ESL_and_HLS.pdf].

EDA for Architecture-Based Design

- Virtual System Prototypes (VSP): Initial offering (VCC, Virtual Component Codesign, Cadence 2000-2002, no success.
- Gradually good HW models (with IP content!) are becoming hard currency. Newer tools (post System C) include ...
  - Synopsys Platform Architect (prev. CoWare),
    ARM Realview SoC Designer/ESL,
    Synopsys CoMET-METeor (prev. VaST) (automotive focus),
    Simics from Wind River (prev. Virtutech).

EDA for Customized Processors



Source: Synopsys ASIP Designer

- Profiles application SW and configures processor to speed up execution. Origin in ARC (1999, ARC 3) and Tensilica (2000).
  - Tensilica Xtensa XPRES (now Synopsys), ASIP Designer (Synopsys, prev. Target Compiler Tech.), ARC (Argonaut RISC Core) (part of Synopsys DesignWare).
  - Read more [Customizable Embedded Processors] on CHANS.

IP (Intellectual Property)

- "Intellectual Property" (IP) cores are subcomponents:
  - from third party or from the company's repository (legacy IP).
  - design (synthesizable) and/or verification IP.
  - HDL description = soft macro (process independent).
  - Physical implementation = hard macro (process dependent).
- Reuse: Saves design/verification time and improves quality.
  - "Consider 80% of a new design is made up of reused HW/SW parts" [Mike Gianfagna, Atrenta].
- Discrete ICs are today insignificant: Fabless CEVA dominates DSP market while fabless ARM dominates uP market.

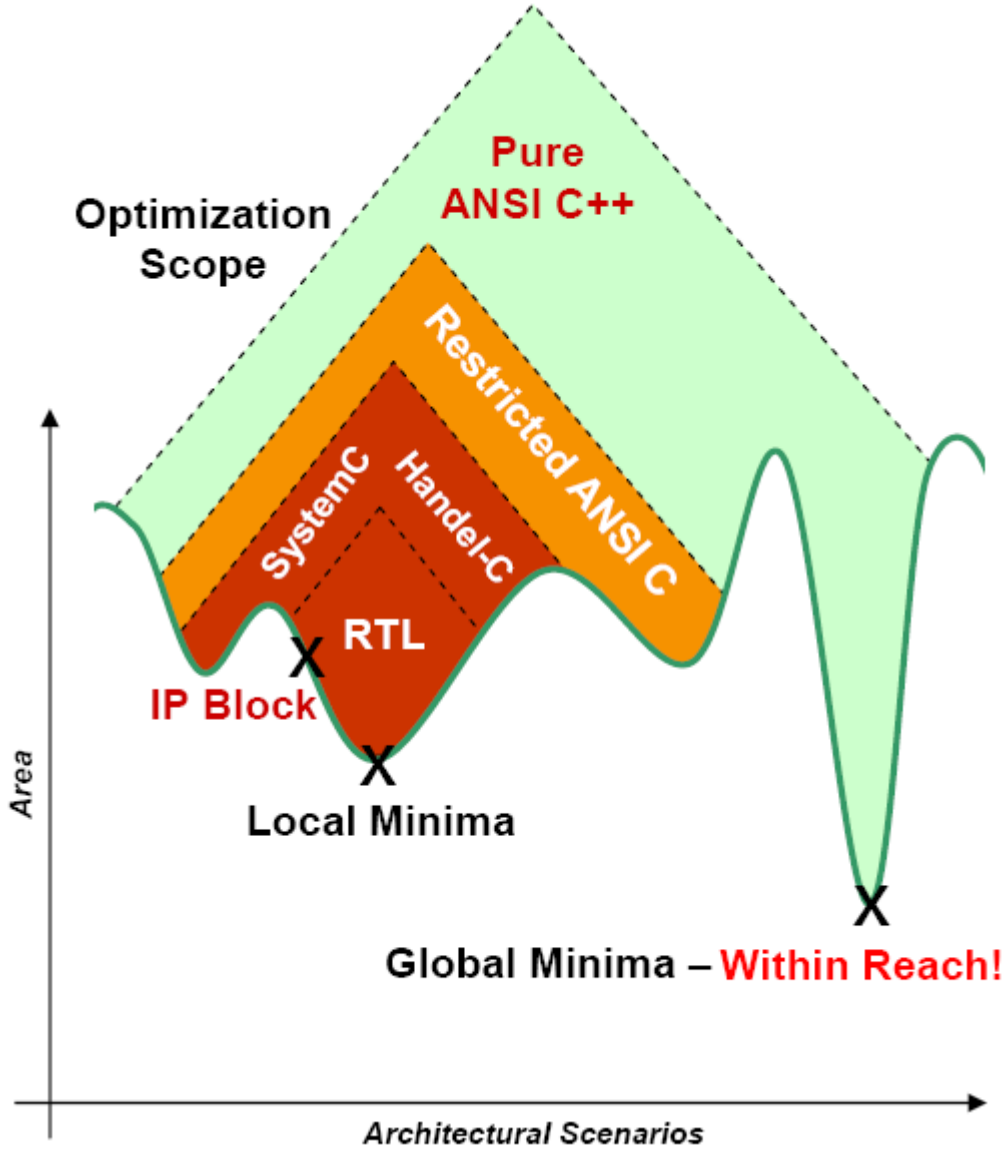
Abstract ESL: What Design Language?

- ANSI C.
  - Standard, which means stable code exists.
  - No concept of time, but think sequentially: Algorithms and dataflows.
  - Proprietary extensions required to handle complex control.
  - Short simulation time.
- System C.
  - C++ class library.
  - Time is present, so think HW:
    Control-dominated applications, such as SoC buses.
  - Slow, event-driven simulation.

Evolution

- Tools change over time. Take for example Catapult C;
  - In 2003, Catapult did HLS, that is, conversion of ANSI C into RTL.
  - In 2010, it accepts both ANSI C and System C and can handle cycle-accurate and transaction-level models; capabilities of an ESL tool.
- Expect tools/design environments to continue to change and converge.
- What about language?
  System C suits system integration well, while ANSI C suits dataflows inside IP blocks. Will the languages converge?

One Opinion [STMicroelectronics DAC'08]



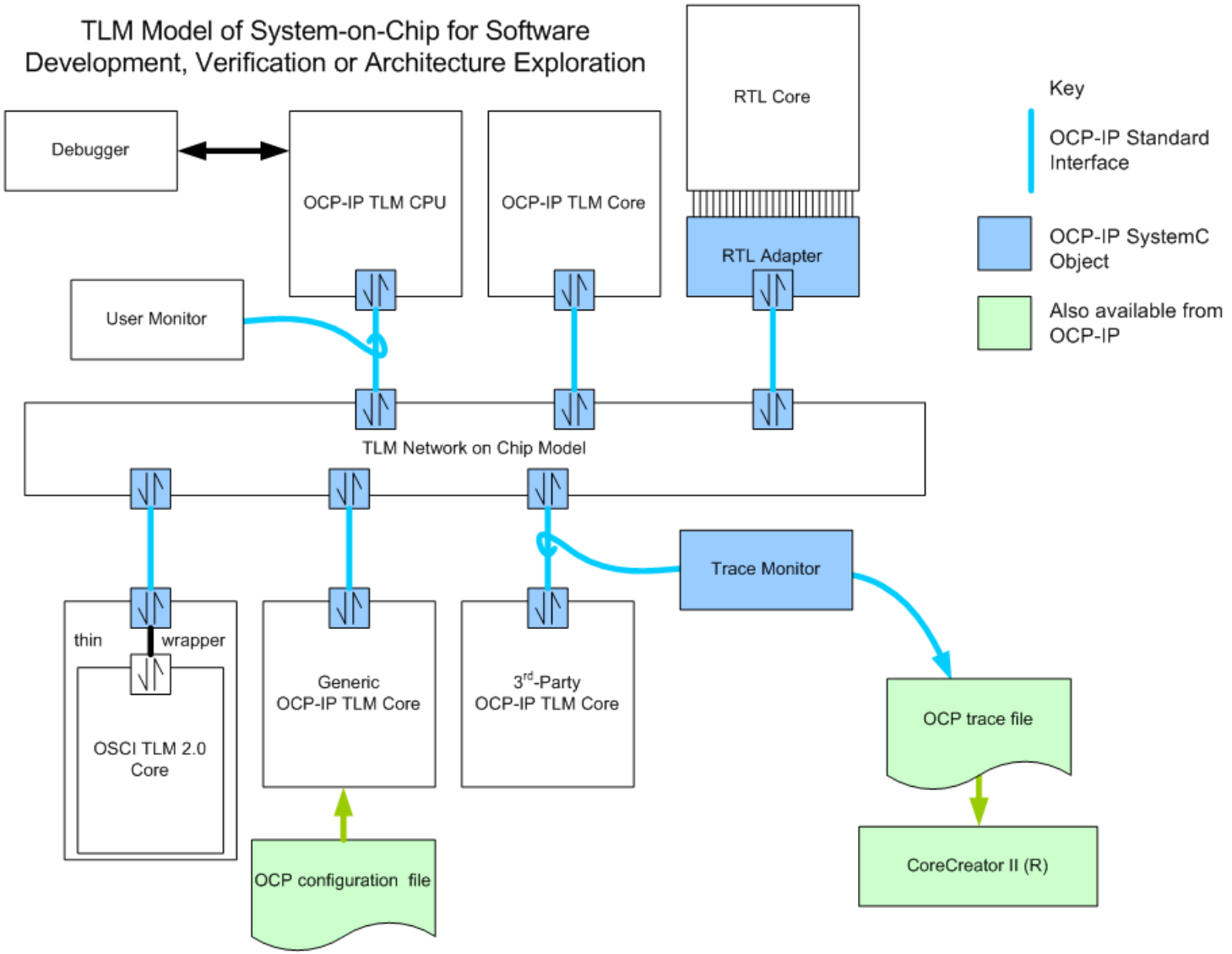
Virtual Prototype Model Abstraction

- Transaction-level models (TLM).
  - Bus transactions.
  - Fast, but inaccurate.
- Cycle-accurate models.
  - Per-cycle HW states.
  - Slower, but more accurate.
- Models for event-driven simulation.
  - Details within the clock cycle.
  - Slowest, but with the highest accuracy.

Example of Popular Model: TLM 2.0

- TLM 2.0 is a System C-based model. Four levels, including ...
  - loosely timed / programmers view: Transaction begins and ends.
  - approximately timed: Transaction contains four timing points.
- Loosely-timed models:
  Processors (such as ARM and IBM),
  interface IPs (such as USB 2/3, AMBA circuits, ...),
  periphery units (such as DMA controller, UART, watchdog, ...).
- Cycle-accurate models exist too:
  For example, IBM PowerPC 405 and 440
  have such support in DesignWare IP library (Synopsys).

Using TLM for System Modeling



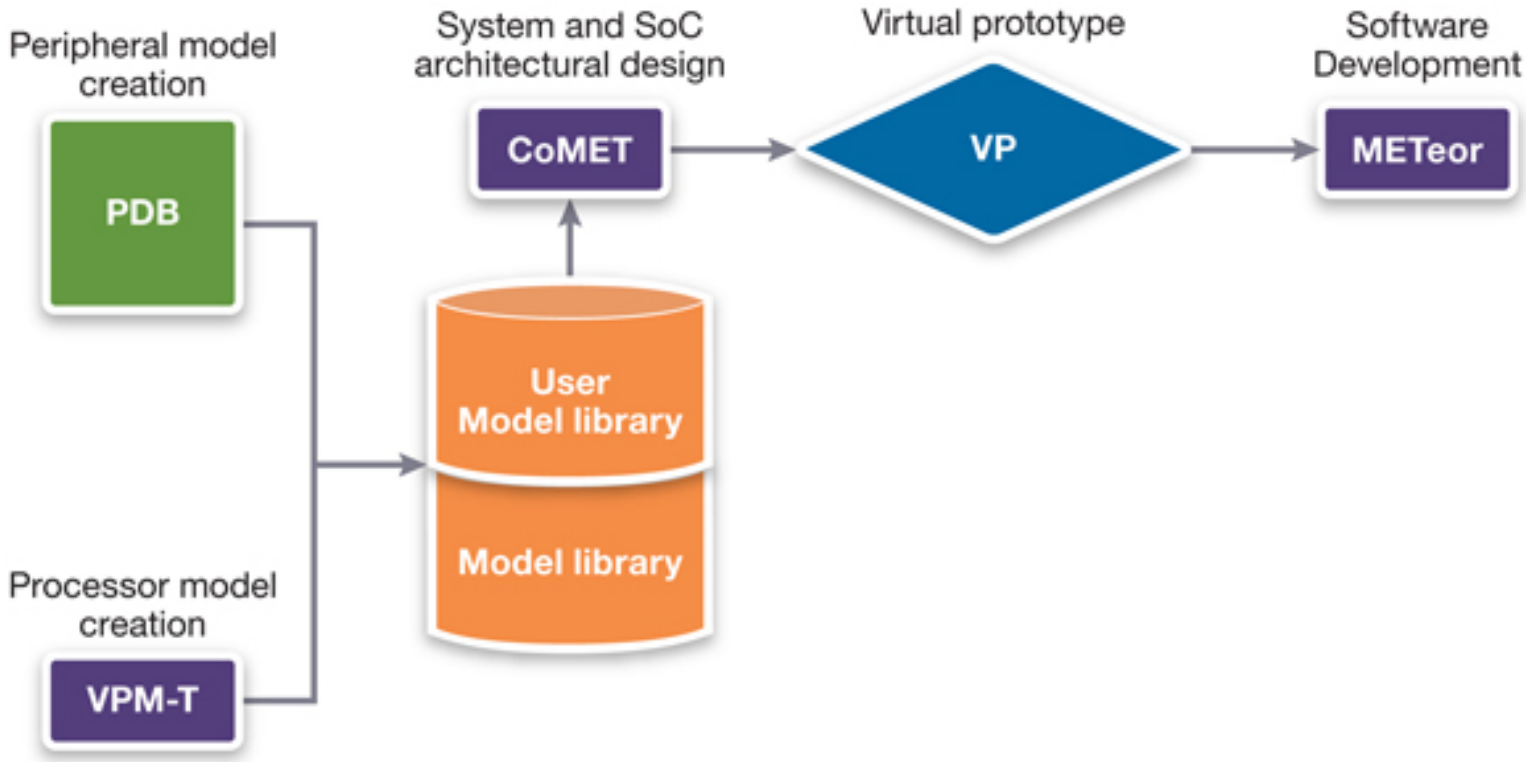
Source: OCP-IP

Complete System Design Flow

- System specification.
- HW/SW partitioning.
- Transaction-level design and verification.
- ESL synthesis to RTL + verification.
- Backend flow RTL -> GDSII.

Tool Example from Synopsys

- A user can create virtual prototype models of periphery HW (interrupt controllers, DMA controllers, UARTs, timers, ...) and integrate those into a platform, for which SW is developed.



uP Virtual Prototypes

- Many different uP models are offered (example from Synopsys).

IP Supplier	Device	Availability
ARM	ARM7TDMI	Available
	ARM926	Available
	ARM946	Available
	ARM968	Available
	ARM1136	Available
	ARM1156	Available
	ARM1176	Available
	MPCore	Available
CEVA	Teaklite	Available
Freescale	e200z6	Available
	eTPU	Available
	e200z3	Available
Intel	XScale	Available

IP Supplier	Device	Availability
NEC	NECV850	Available
	NECV850ES	Available
	NECV850E1	Available
Renesas	SH2A	Available
	R32C	Available
	H8SX	Available
StarCore	SC1400	Available
	SC1200	Available
	SC3400	Available
	SC2400	Available
Toshiba	TX99	Available
	TX19A	Available
	TX49H3	Available

Electronic Systems and EDA: Conclusion

- Electronic systems grow more complex $\Rightarrow$
  need for new, higher abstractions to maintain productivity.
  - Complex systems also mean complex verification.
  - SoCs integrate several processors $\Rightarrow$ SW and HW must be simultaneously developed. Virtual prototypes?
- EDA rapidly developing:
  - EDA frameworks made up of disparate parts. Stay alert ! Design flows: How to navigate through framework is essential.
  - many mergers and acquisitions: *http://www.semiwiki.com/forum/showwiki.php?title=Semi+Wiki:EDA+Mergers+and+Acquisitions+Wiki*