Robotics and Embedded Systems

Technische Universität München

Design Methodologies

Kai Huang

Outline

- Design Trend Recap
- Gajski's Y-Chart
- Kienhuis Y-Chart
- Model-based Design

Embedded Systems Design

- Embedded Systems Design is NOT just a special case of either hardware (Computer/Electrical Engineering) or software (Software Engineering/Computer Science) design.

- It has functional requirements (expected services), and it has non-functional requirements /constraints
  - Interaction constraints: deadlines, throughput, jitter
  - Execution constraints: available resources, power, failure rates

- Embedded Systems design discipline needs to combine
  - Computer Science
  - Computer/Electrical Engineering

Trends in Embedded Systems

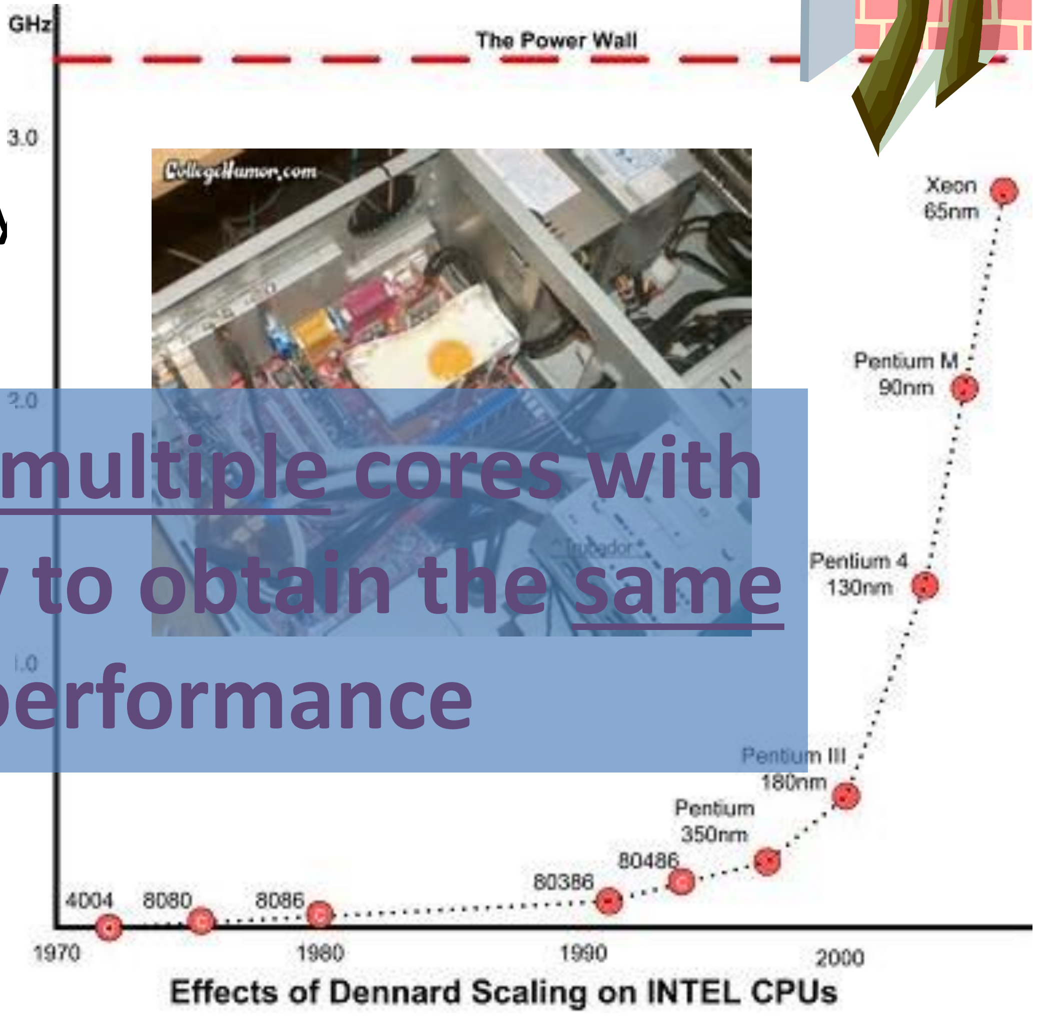
- Higher Degree of Integration
  - Moore's law

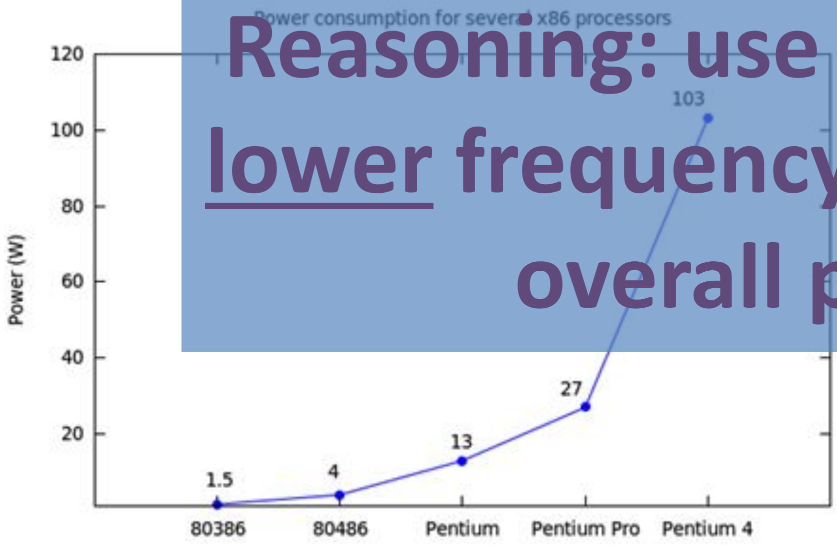
- Power wall
  - Towards Multi-Processor (System-on-Chip)

- Software Increasing
  - Flexibility and time-to-market

Power Wall

- Law of Physics: All electrical power consumed is eventually radiated as heat





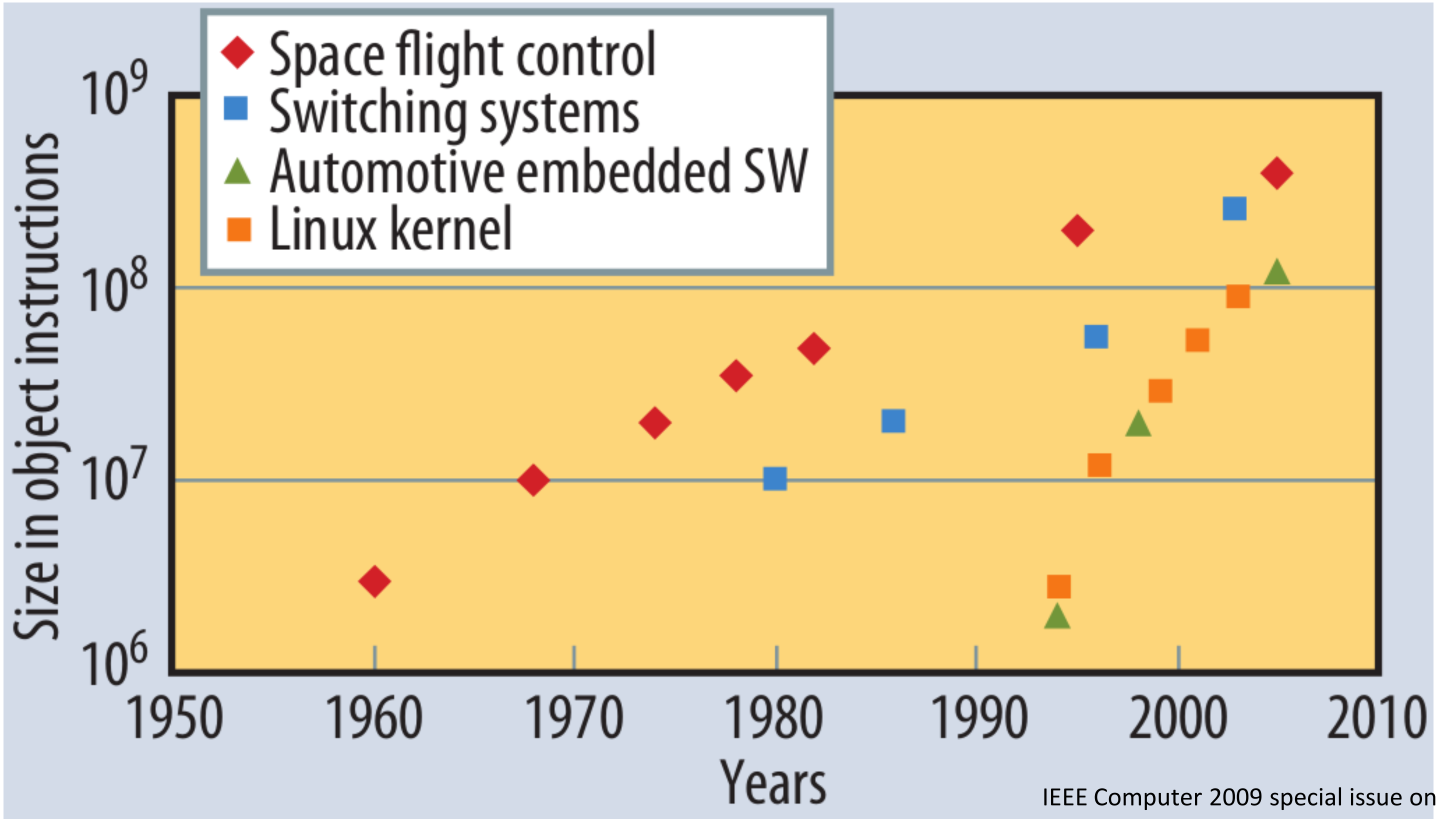
Reasoning: use <u>multiple</u> cores with <u>lower</u> frequency to obtain the <u>same</u> overall performance

Power Wall for MPSoC

Reasoning: packing more transistors needs deeper sub micro CMOS techniques which results in larger leakage current

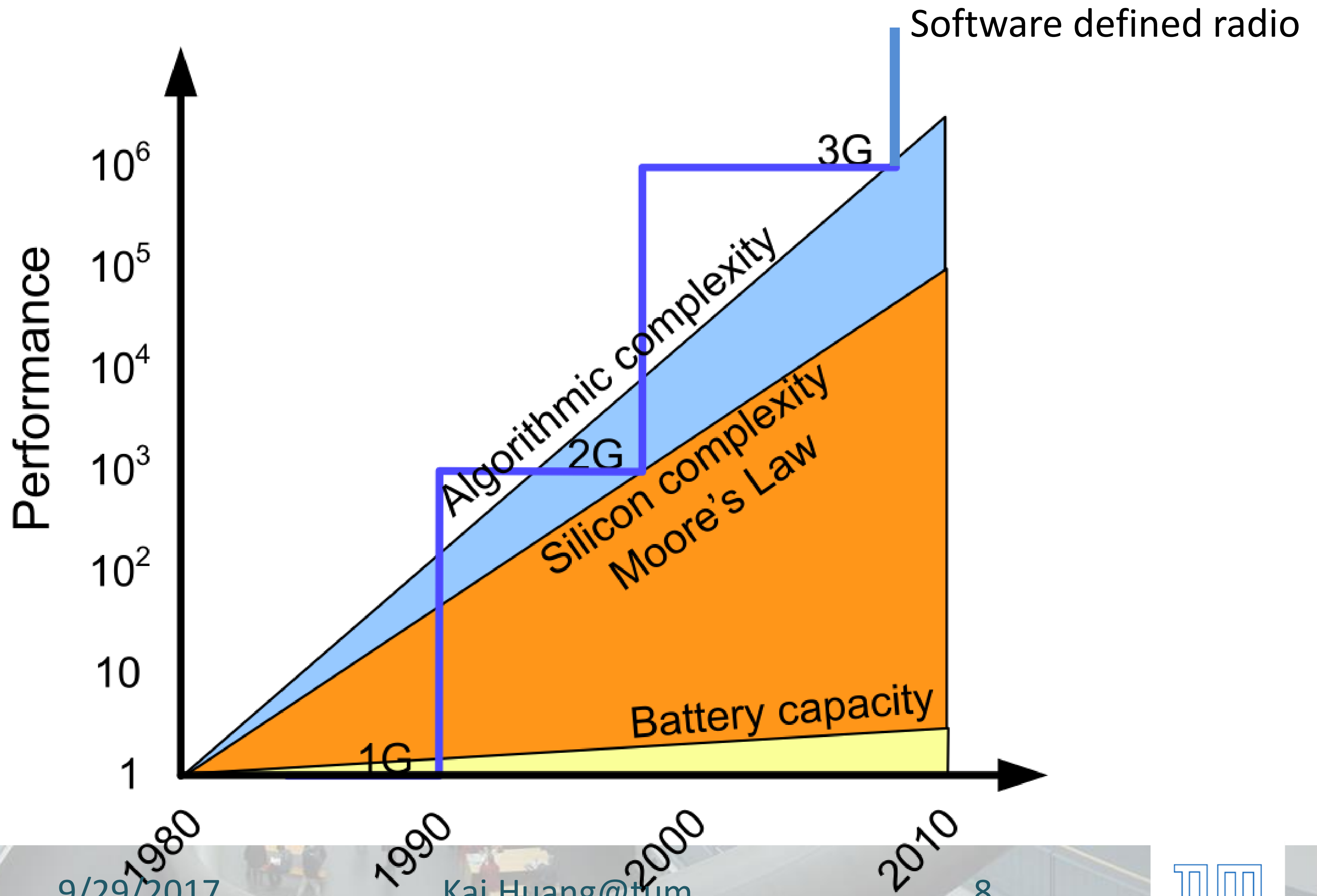
Embedded Software Complexity

- Software engineers always push the limits of the hardware capability



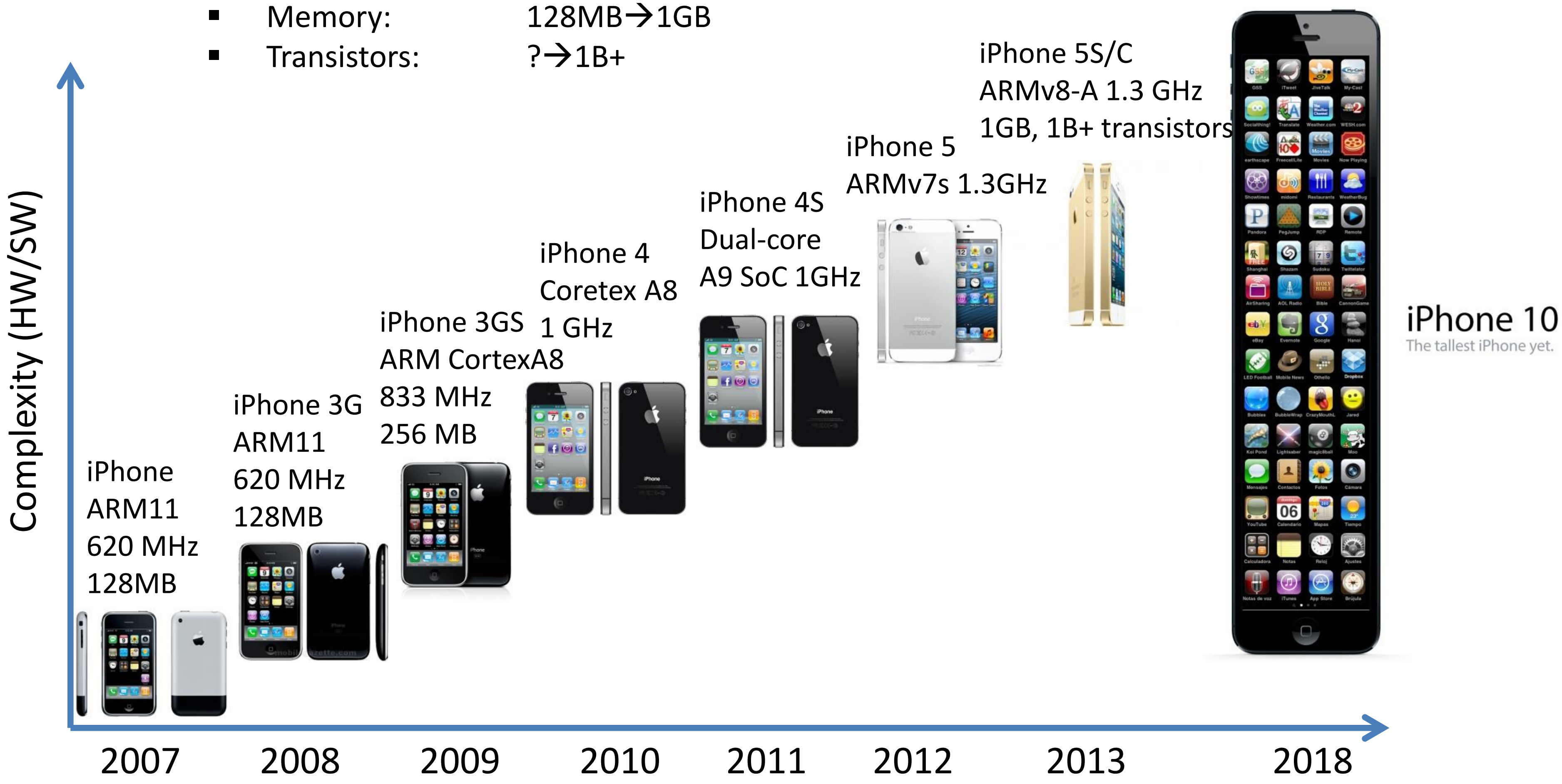
IEEE Computer 2009 special issue on Embedded Software

Telecom Example



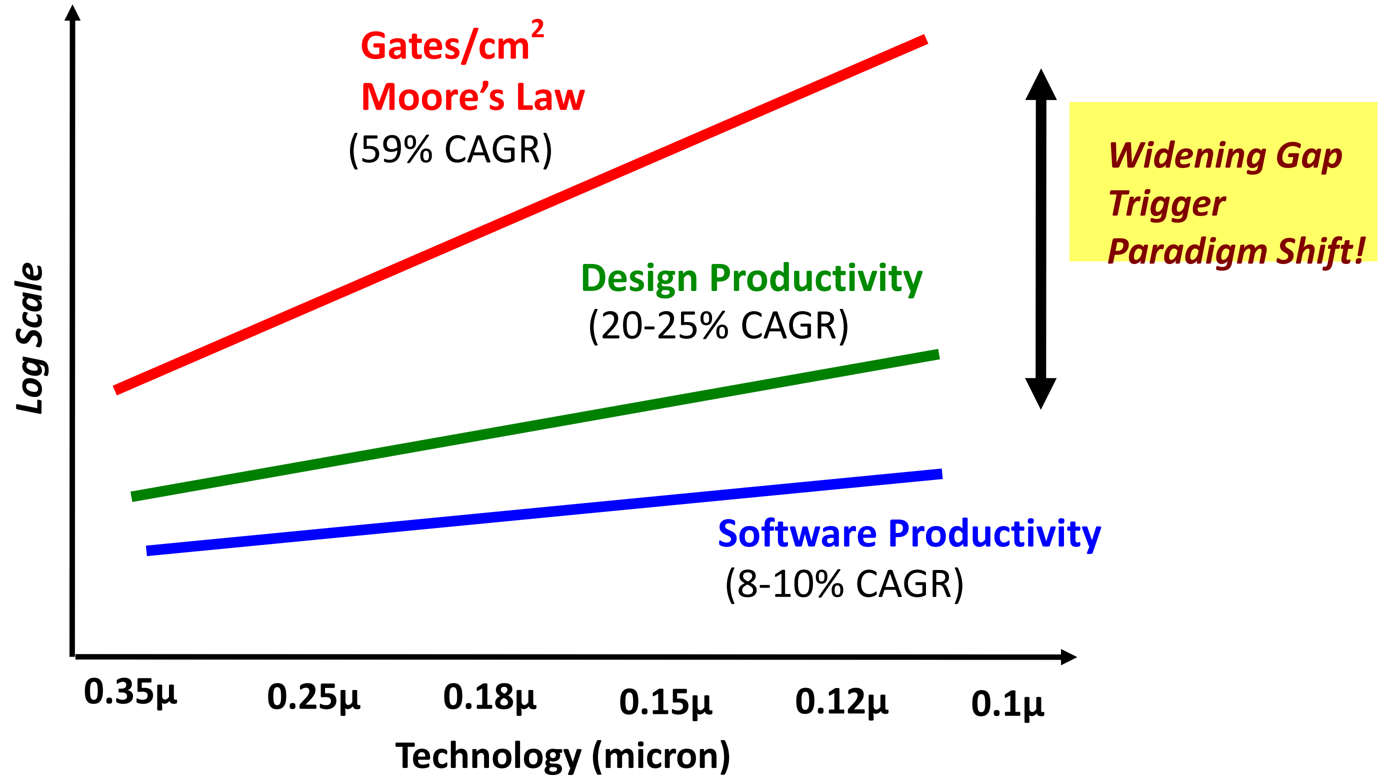
iPhone

- Frequency: 620MHz→1.3GHz
- Memory: 128MB→1GB
- Transistors: ?→1B+



Data from: http://en.m.wikipedia.org/wiki/Apple_(system_on_chip)

Design Crisis: Design Productivity Gap



- The well-know productivity gap generated by the disparity between the rapid paces the design complexity increased in comparison to that of design productivity

CAGR: Compound Annual Growth Rate

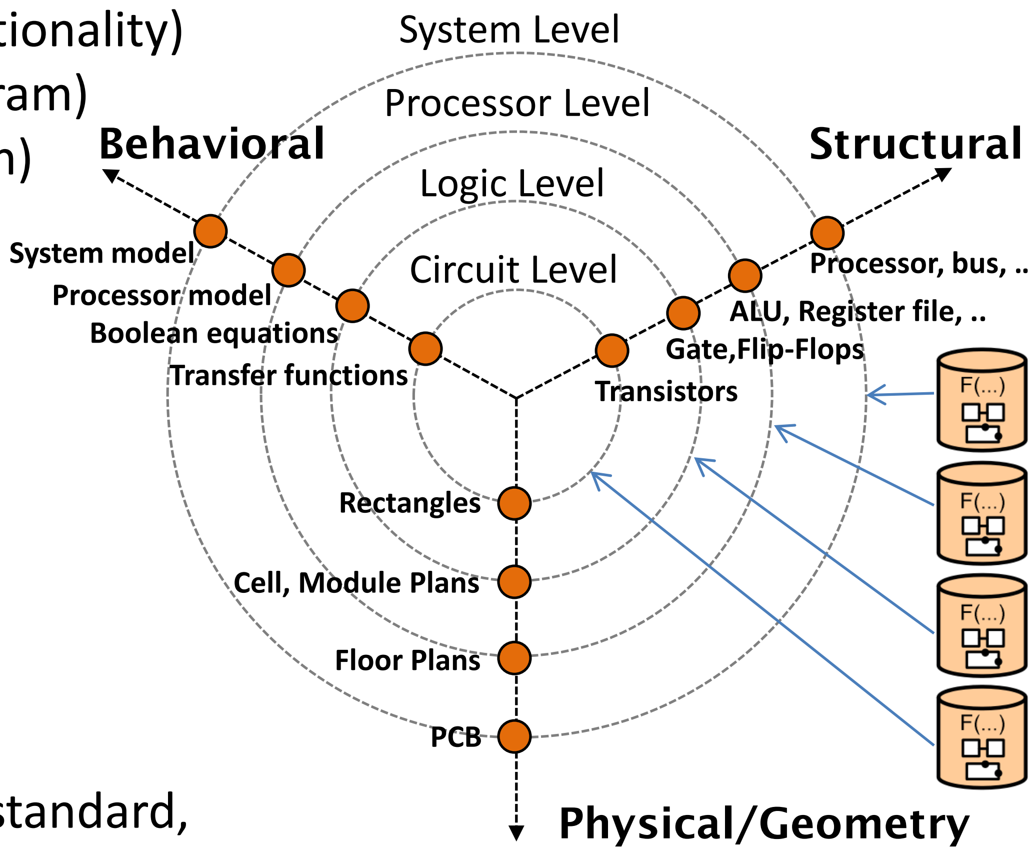
Needs of New Methodology

- Kurt Keutzer, et. al. “System-Level Design: Orthogonalization of Concerns and Platform-Based Design," IEEE TCAD, 19(12), December 2000.
  - Google citation: 1016

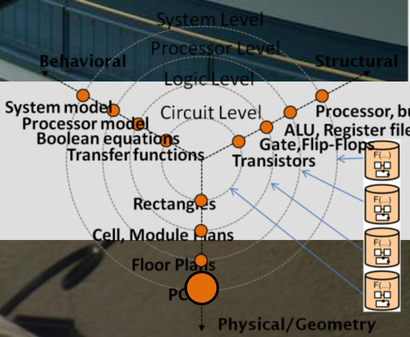
“we believe that the lack of appropriate methodology and tool support for modeling of concurrency in its various forms is an essential limiting factor in the use of both RTL and commonly used programming languages to express design complexity”

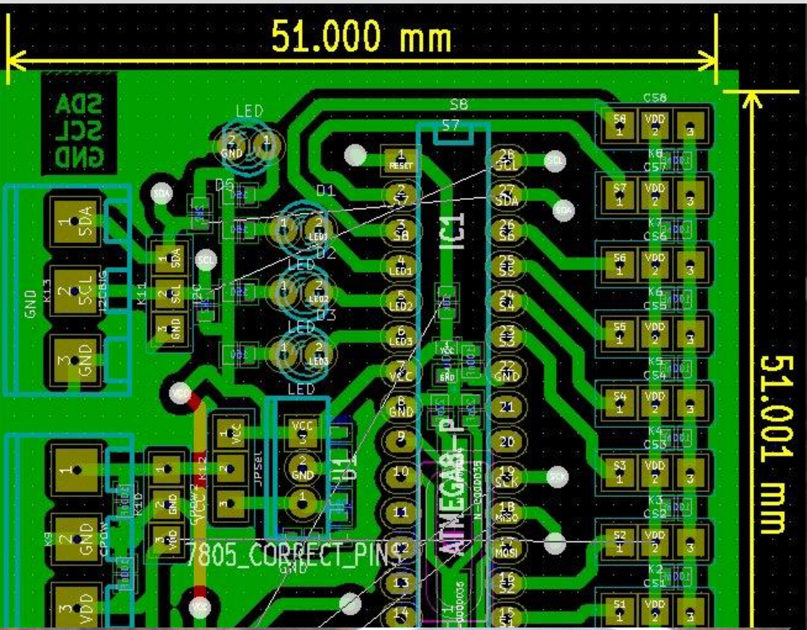
System Design: Gajski Y-Chart

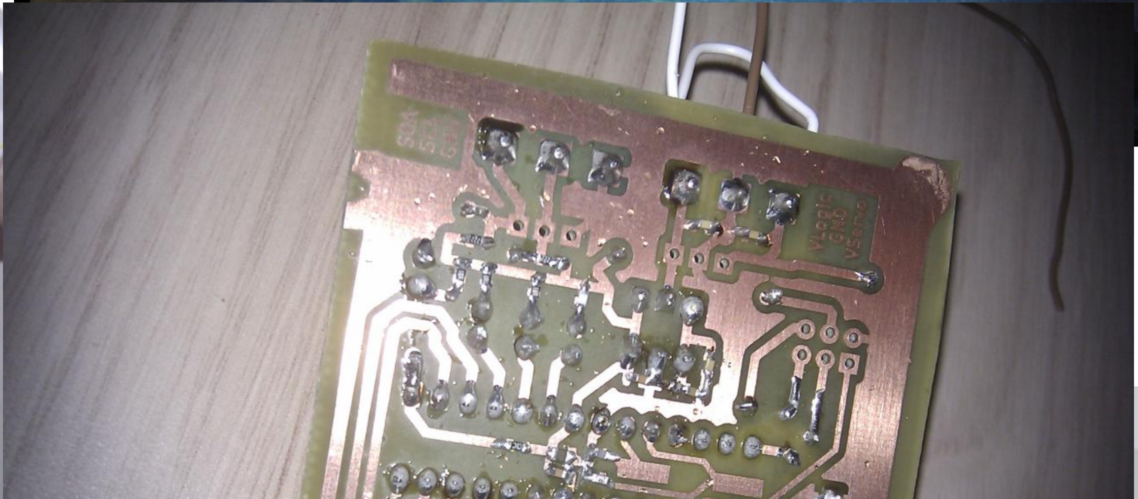
- Three design views
  - Behavior (specification/functionality)
  - Structure (netlist/block diagram)
  - Physical (layout/board design)
- Four abstraction levels
  - Circuit level
  - Logic level
  - Processor (RTL) level
  - System level
- Four component libraries
  - Transistors
  - Logic (standard cells)
  - RTL (ALUs, RFs, ...)
  - Processor/Communication (standard, custom)



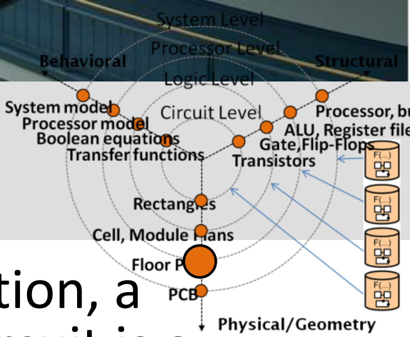
Printed Circuit Board (PCB)



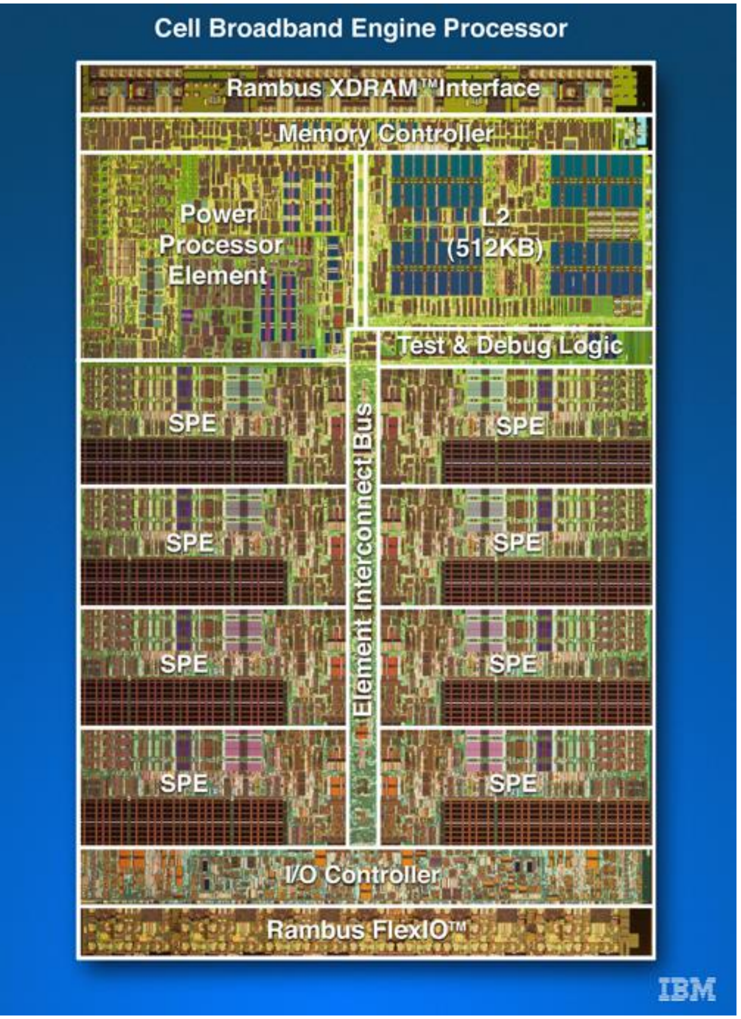


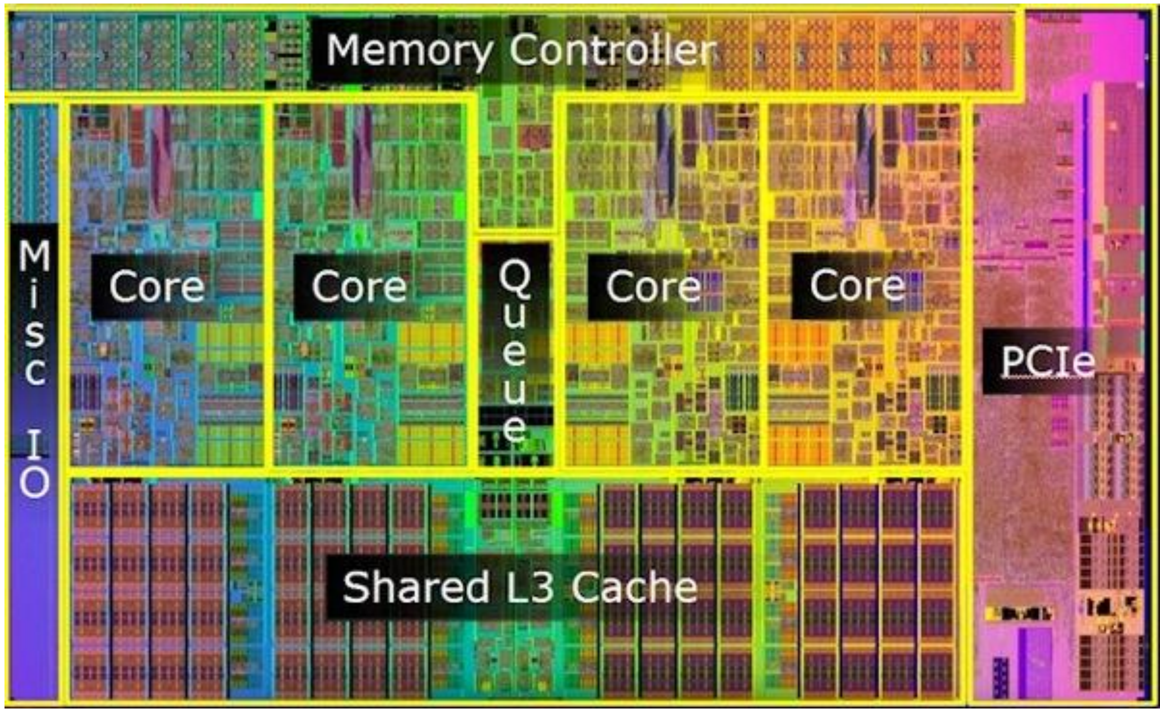


Floorplan



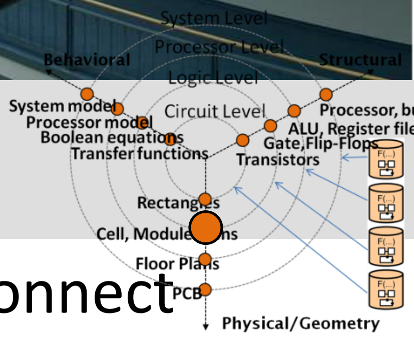
- In electronic design automation, a floorplan of an integrated circuit is a schematic representation of tentative placement of its major functional blocks.



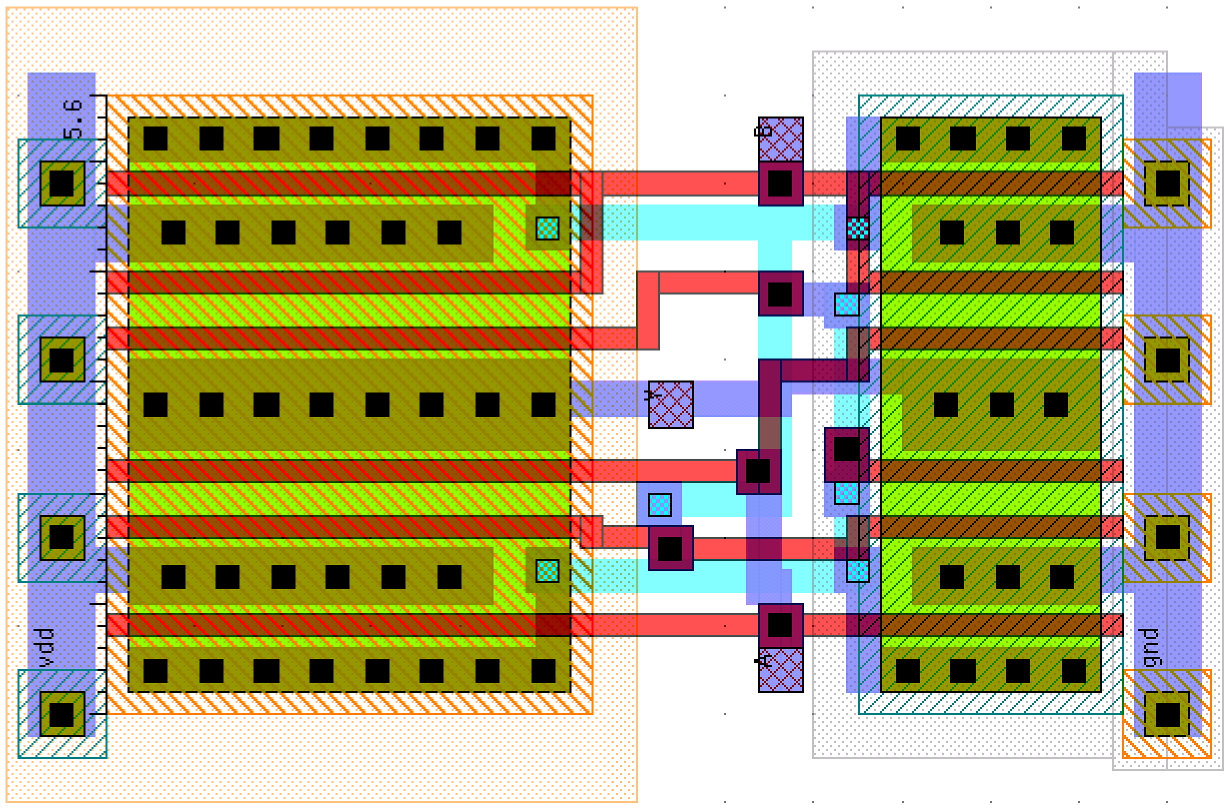


Intel Lynnfield (Core i5/i7)

Standard Cell

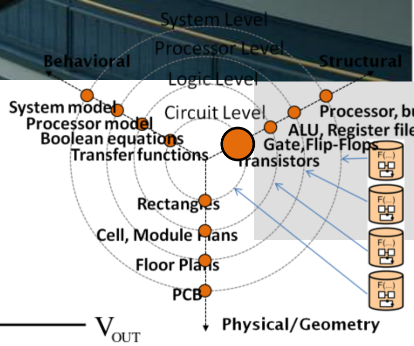


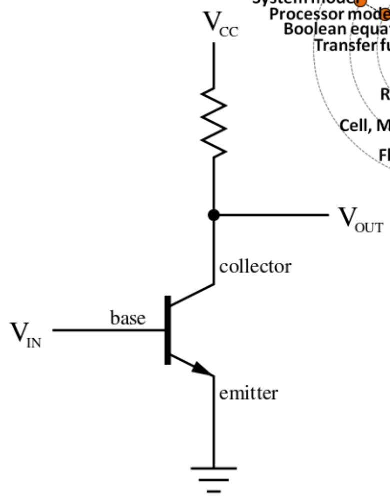
- A standard cell is a group of transistor and interconnect structures that provides a Boolean logic function or a storage function

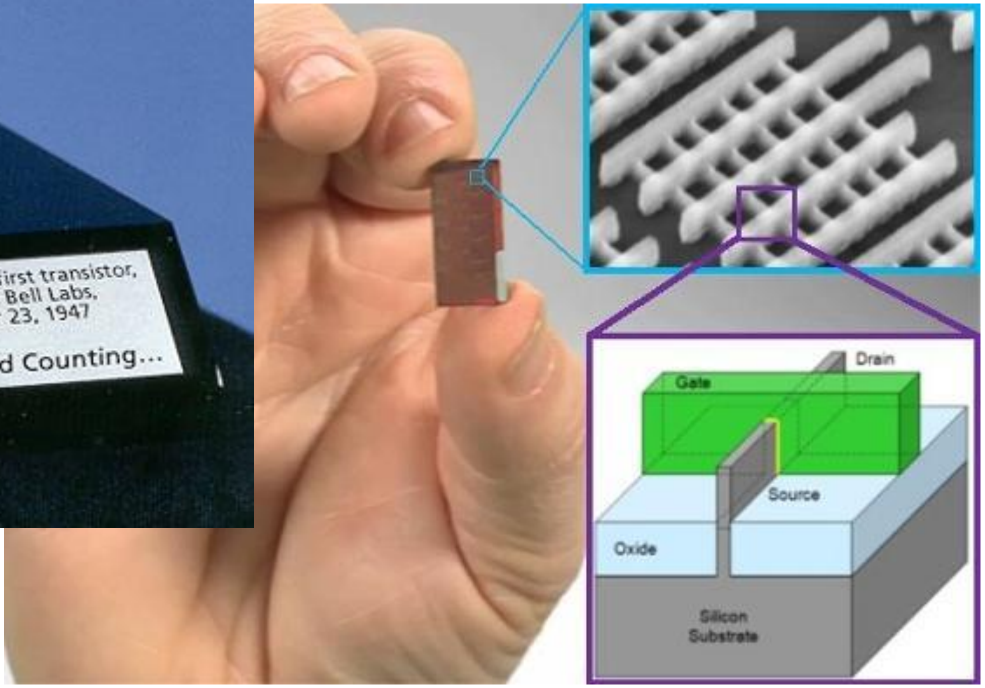


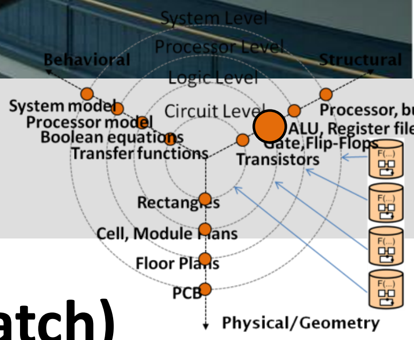
IIT/OSU standard cell library 2-XOR gate in 0.18µm technology

Transistors



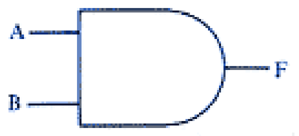
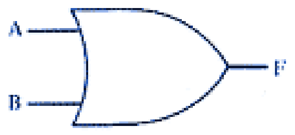
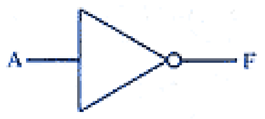
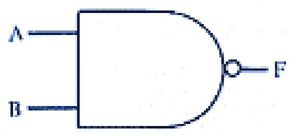
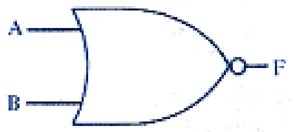




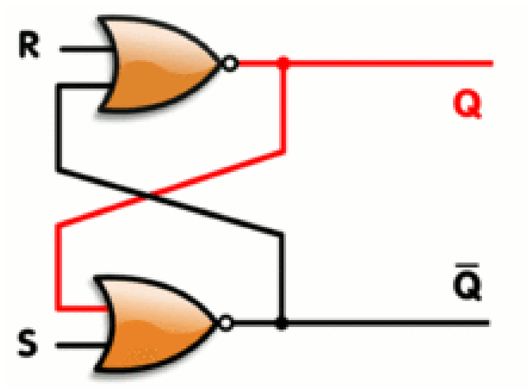


Logic

Gate

Name	Graphic Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \bar{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{(AB)}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{(A + B)}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

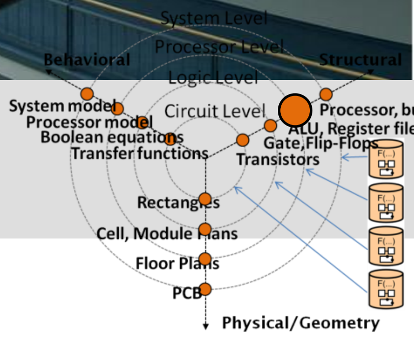
Flip-Flop (SR NOR latch)

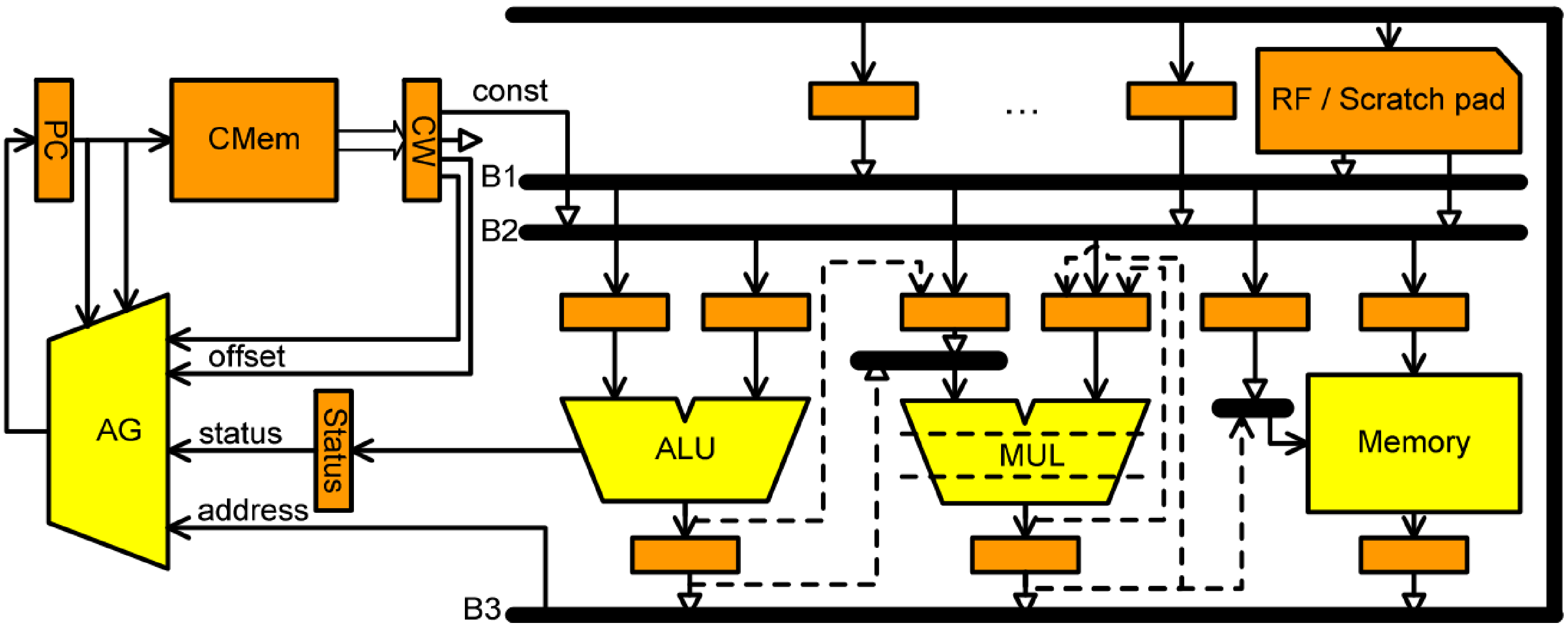


Characteristic table			
S	R	Q_{next}	Action
0	0	Q	hold state
0	1	0	reset
1	0	1	set
1	1	X	not allowed

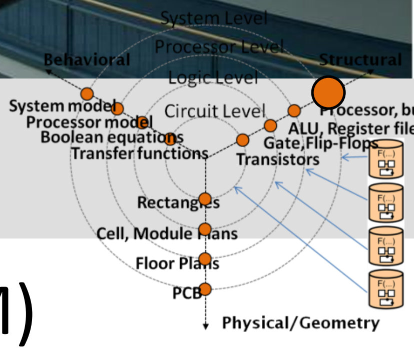
- where S and R stand for *set* and *reset*

Processor Structure Model

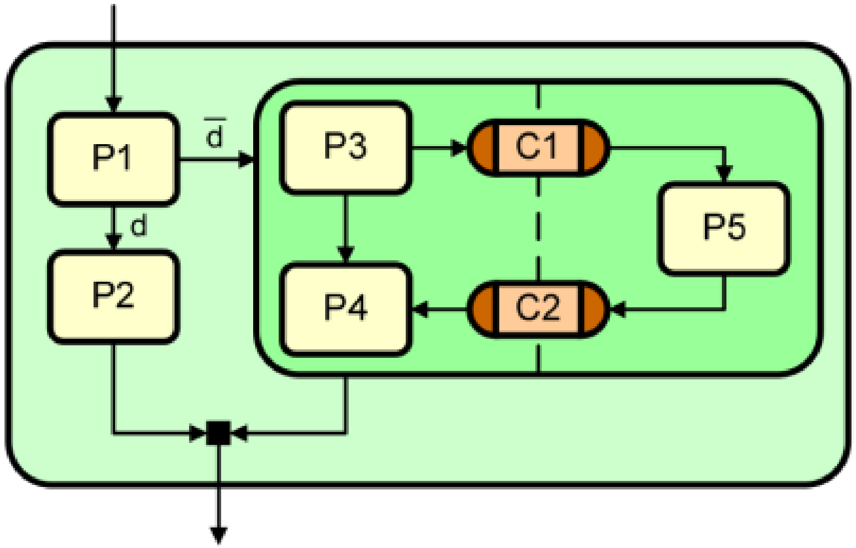


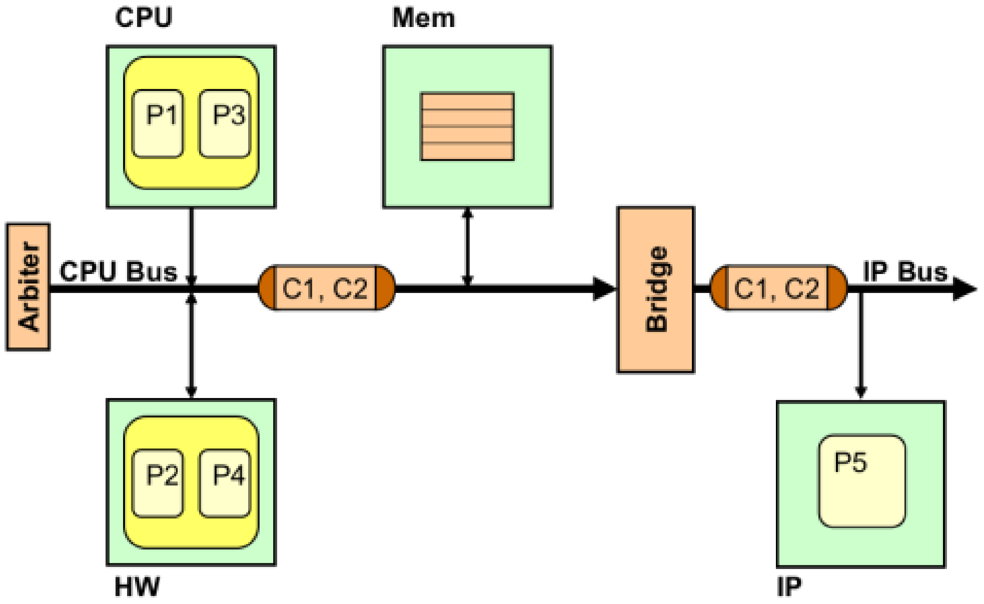


System Model



- Behavior (MoC)
- Structure (TLM)

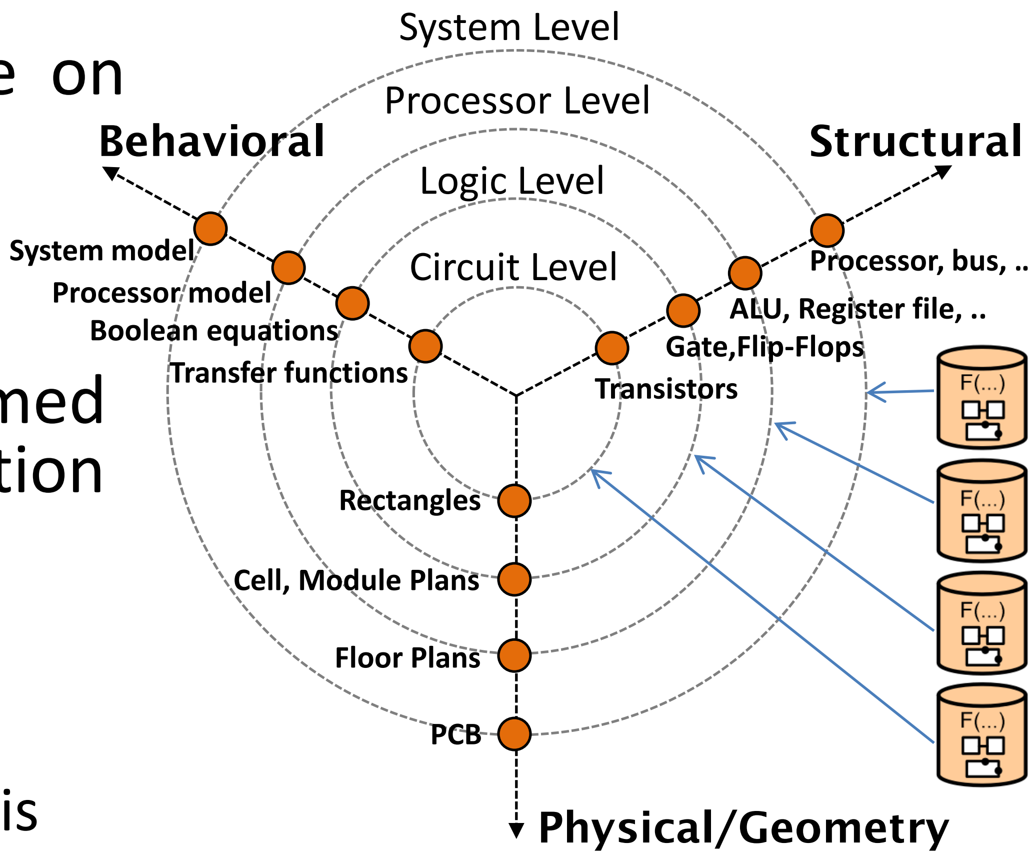




9/29/2017 Kai.Huang@tum 19

Synthesis

- Definition: The process of converting the given behavior into a structure on an abstraction level

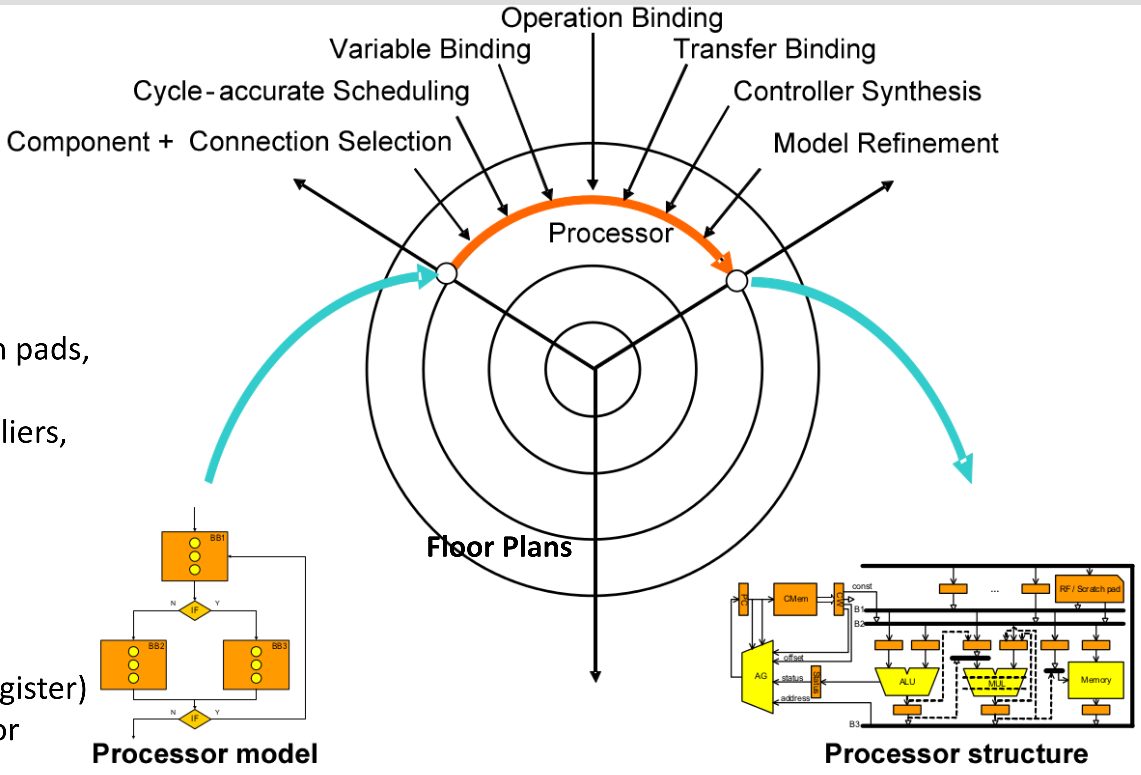


- Synthesis can be performed at every level of abstraction

- Examples:
  - Processor Level Synthesis
  - System Level Synthesis

Processor Level Synthesis

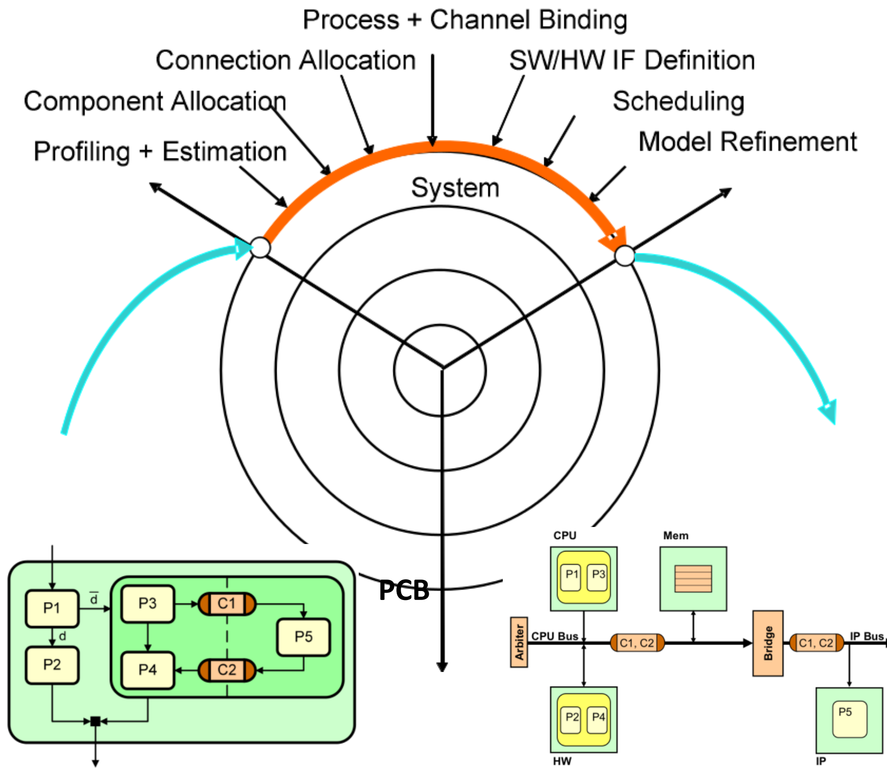
- Processor model
  - FSM with Datapath
  - CDFG
  - Instruction set flow chart
- Processor structure model
  - Datapath components
    - Storage (registers, RFs, Scratch pads, data memories)
    - Functional units (ALUs, multipliers, shifters, special functions)
    - Connection (buses, selectors, bridges)
  - Controller components
    - Registers (PC, Status register, Control word or Instruction register)
    - Others (AG, Control memory or Program memory)
  - Processor structure
    - Pipelining, chaining, multi-cycling, forwarding



- Synthesis consists of several tasks: many different sequences possible
  - Different models, different libraries, different features, different structures
  - Different tools, different metrics, different quality

System Level Synthesis

- System behavior model
  - Use a MoC
  - Many MoCs exist
- System structural model
  - Set of computational components
    - Processors
    - IPs
    - Custom HW components
    - Memories
  - Set of communication components
    - Buses, bridges, arbiters
    - NoCs

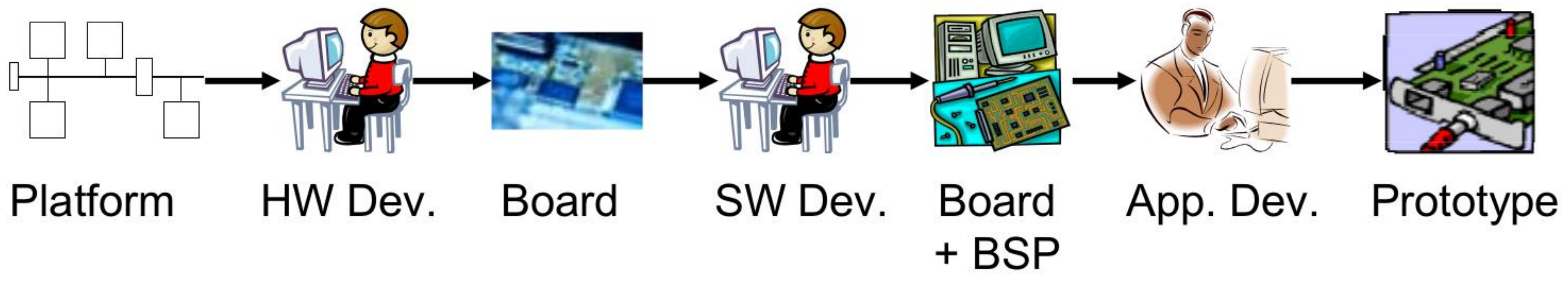


- Synthesis consists of several tasks: different sequences possible
  - Different MoCs, different libraries, different features, different platforms
  - Different tools, different metrics, different quality

Design Flows (Gajski's view)

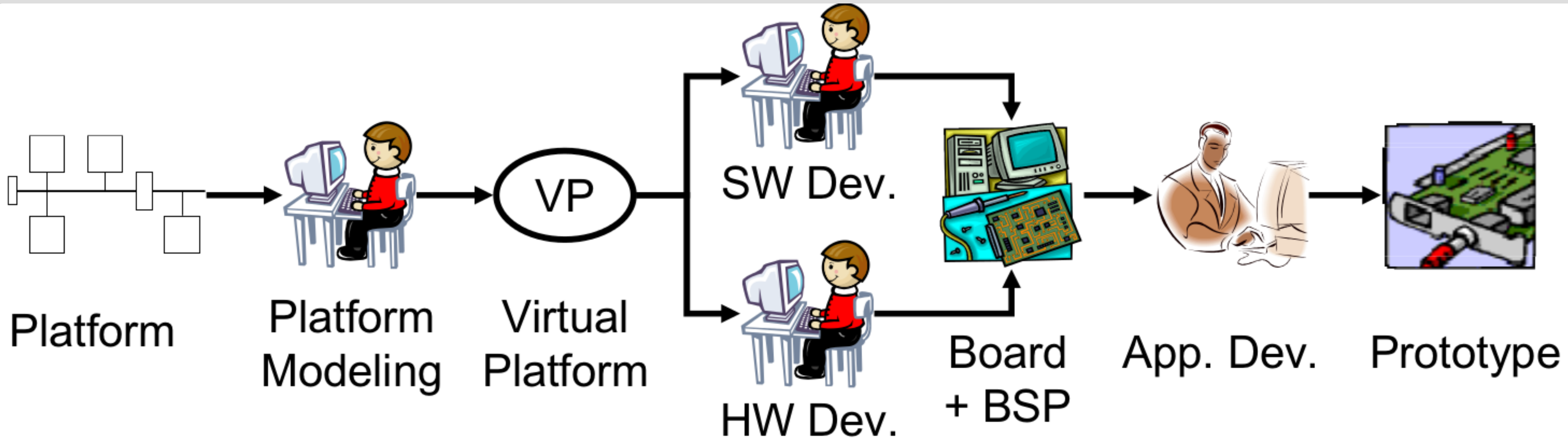
- Three generic evolutionary design flows
  - Capture-and-Simulate (1960s to 1980s)
    - Designers do the complete design manually, no automation
    - Designers validate the design through simulation at the end of the design
  - Describe-and-Synthesize (late 1980s to late 1990s)
    - Designers describe just functionality, tools synthesize structure
    - Simulation before and after the synthesis
  - Specify-Explore-Refine (early 2000 to present)
    - System design performed at several levels of abstraction
    - At each level of abstraction designers:
      - First, specify/model the system under design
      - Then, explore alternative design decisions
      - Finally, refine the model according to their decisions (i.e., put more details)
    - The refined model is used as a specification for the next lower level

Traditional System Design



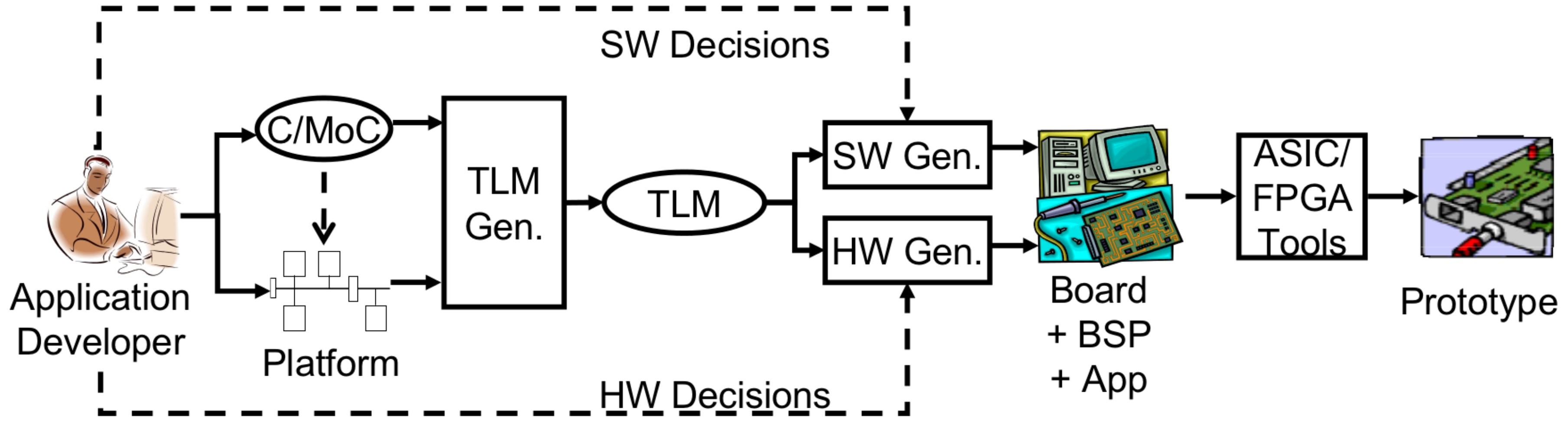
- Hardware first approach
  - Platform is defined by architect or based on legacy
  - Designers develop and verify RTL model of platform
  - Slow error prone process
- SW development after HW is finalized
  - Debugging is complicated on the board due to limited observablity
  - HW errors found during SW development are difficult to rectify
- Application is ported after system SW is finalized

Virtual Platform based System Design



- Virtual platform (VP) is a fast model of the HW platform
  - Typically an instruction set simulator or C/C++ model of the processor
  - Peripherals are modeled as remotely callable functions
  - Executes several orders of magnitude faster than RTL
- SW and HW development are concurrent
  - VP serves as the golden model for both SW and HW development
  - SW development can start earlier
  - HW designers can use SW for realistic test bench for RTL

Model-based System Design



- Model based design gives control to application developers
  - Application is captured as high level C/C++/UML specification
  - Transaction level model (TLM) is used to verify and evaluate the design
- System synthesis
  - The best platform for given application can be synthesized automatically
  - For legacy platforms, application mapping can be generated automatically
  - Cycle accurate SW/HW can be generated from TLM for implementation

Modeling, Design, Analysis

- Modeling is the process of gaining a deeper understanding of a system through imitation. Models specify what a system does.

- Design is the structured creation of artifacts. It specifies how a system does what it does. This includes optimization.

- Analysis is the process of gaining a deeper understanding of a system through dissection. It specifies why a system does what it does (or fails to do what a model says it should do).

What for Modeling?

- Developing insight about a system, process, or artifact through imitation.
- A model is the artifact that imitates the system, process, or artifact of interest.
- A mathematical model is model in the form of a set of definitions and mathematical formulas/objects.

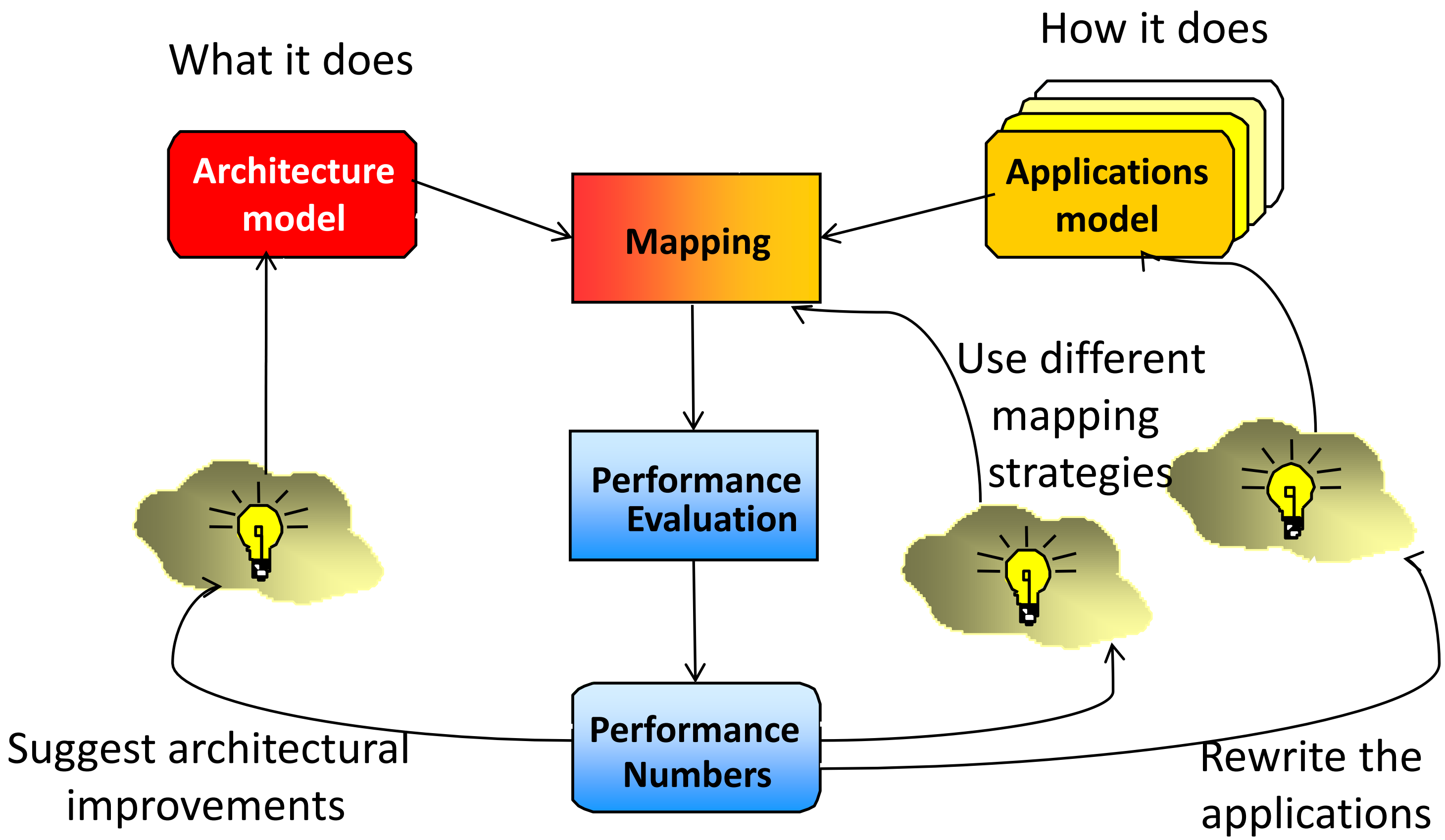
What is Model-Based Design?

- Create a mathematical model of all the parts of the embedded system
  - Physical world
  - Control system
  - Software environment
  - Hardware platform
  - Network
  - Sensors and actuators

$\left.\right\}$ Different sub-systems, different approaches to modeling

- Construct the implementation from the model
  - Goal: automate this construction, like a compiler
  - In practice, only portions are automatically constructed

The Other Y-Chart [Kienhuis et al.]



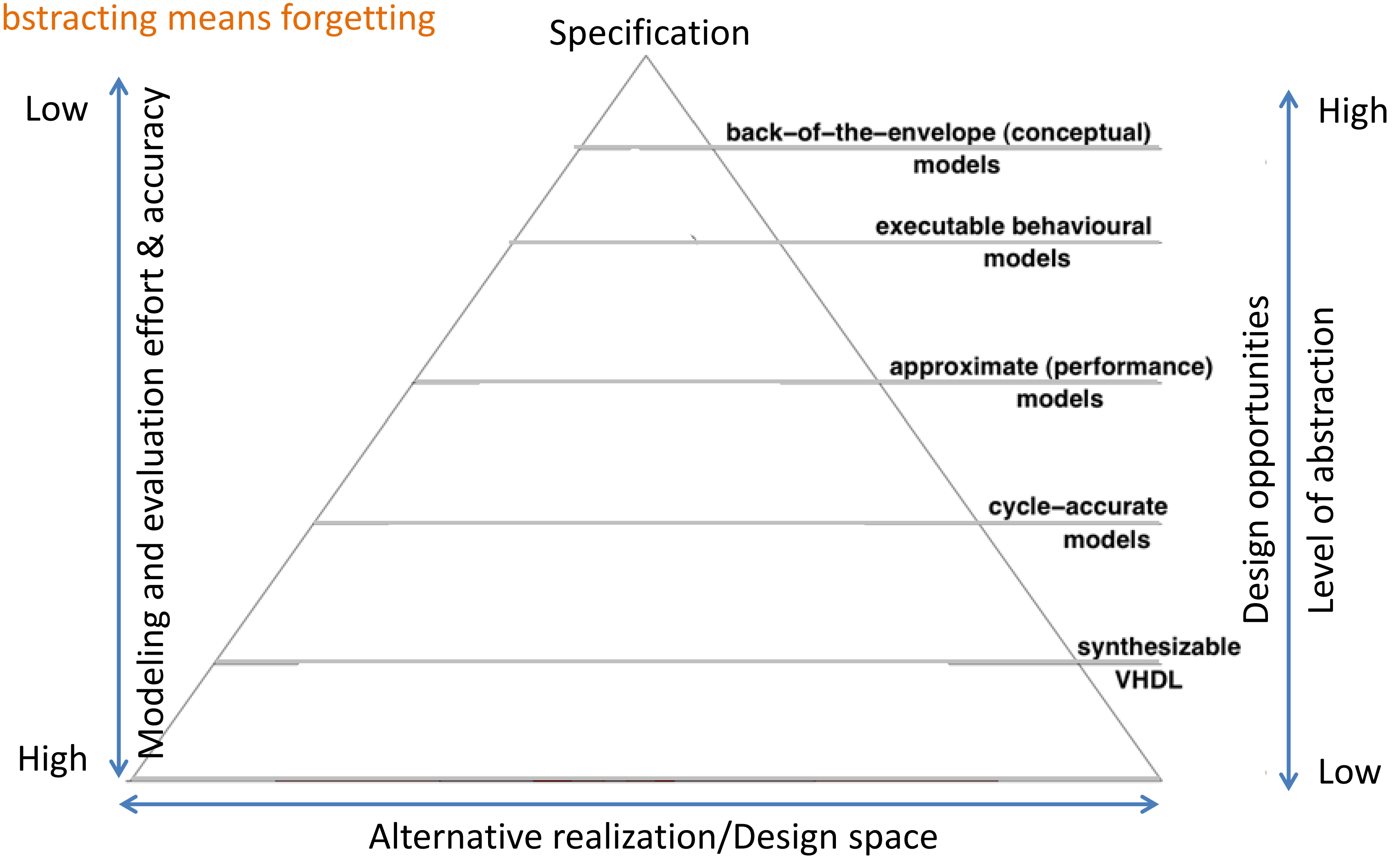
Three different ways to improve the performance of a system

The Other Y-Chart

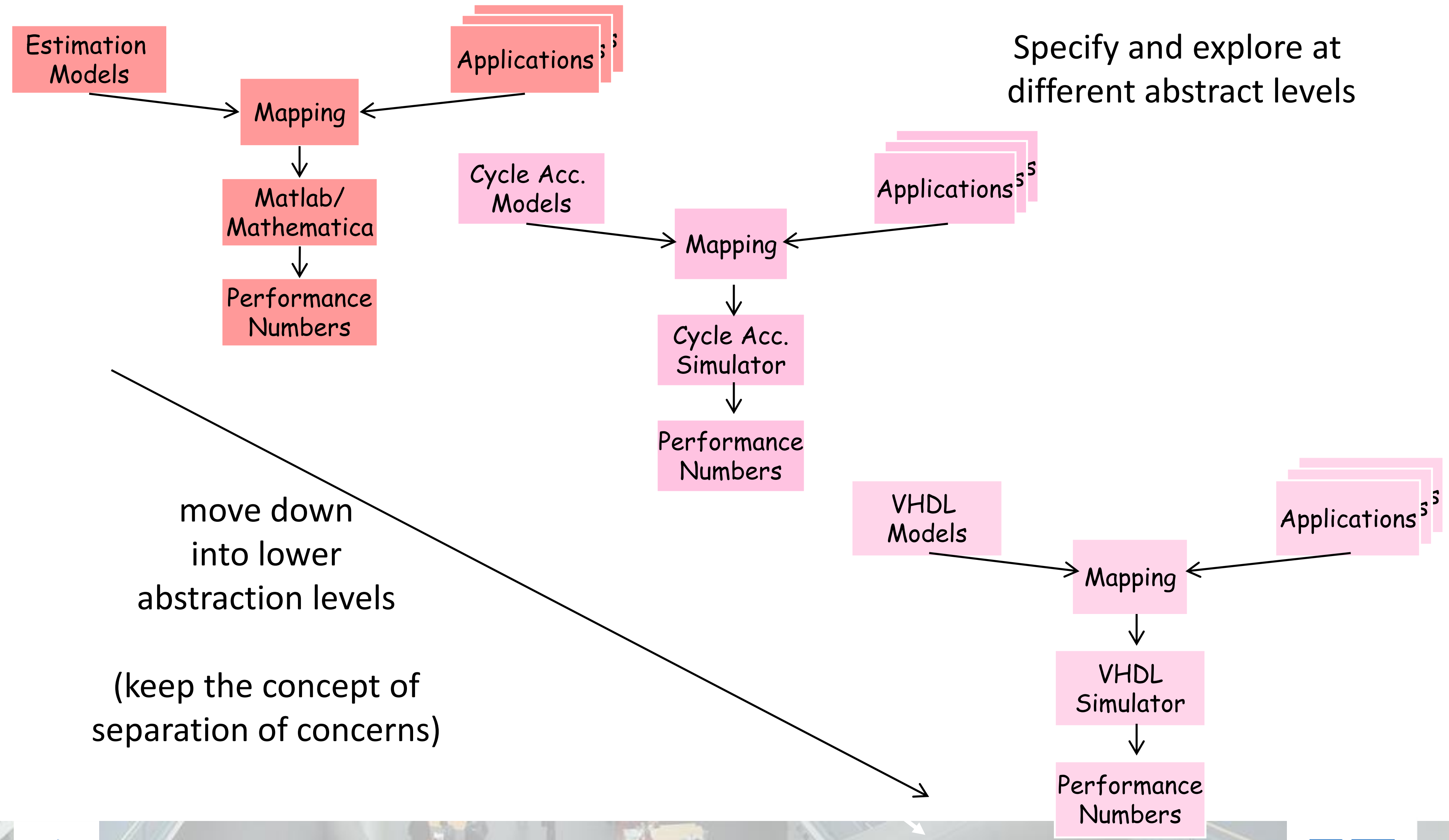
- Separation of Concerns
  - Application vs. architecture modeling
- Different to Gajski Y-Chart
  - Gajski Y-Chart: covers mainly the synthesis aspect
  - Kienhuis Y-Chart: covers mainly the quality assessment aspect

Y-Chart Design BUT at Which Level of Abstraction?

Abstracting means forgetting


Specification
back–of–the–envelope (conceptual) models
executable behavioural models
approximate (performance) models
cycle–accurate models
synthesizable VHDL
Low
High
Modeling and evaluation effort & accuracy
High
Low
Design opportunities
Level of abstraction
Alternative realization/Design space

Stack of Y-Chart

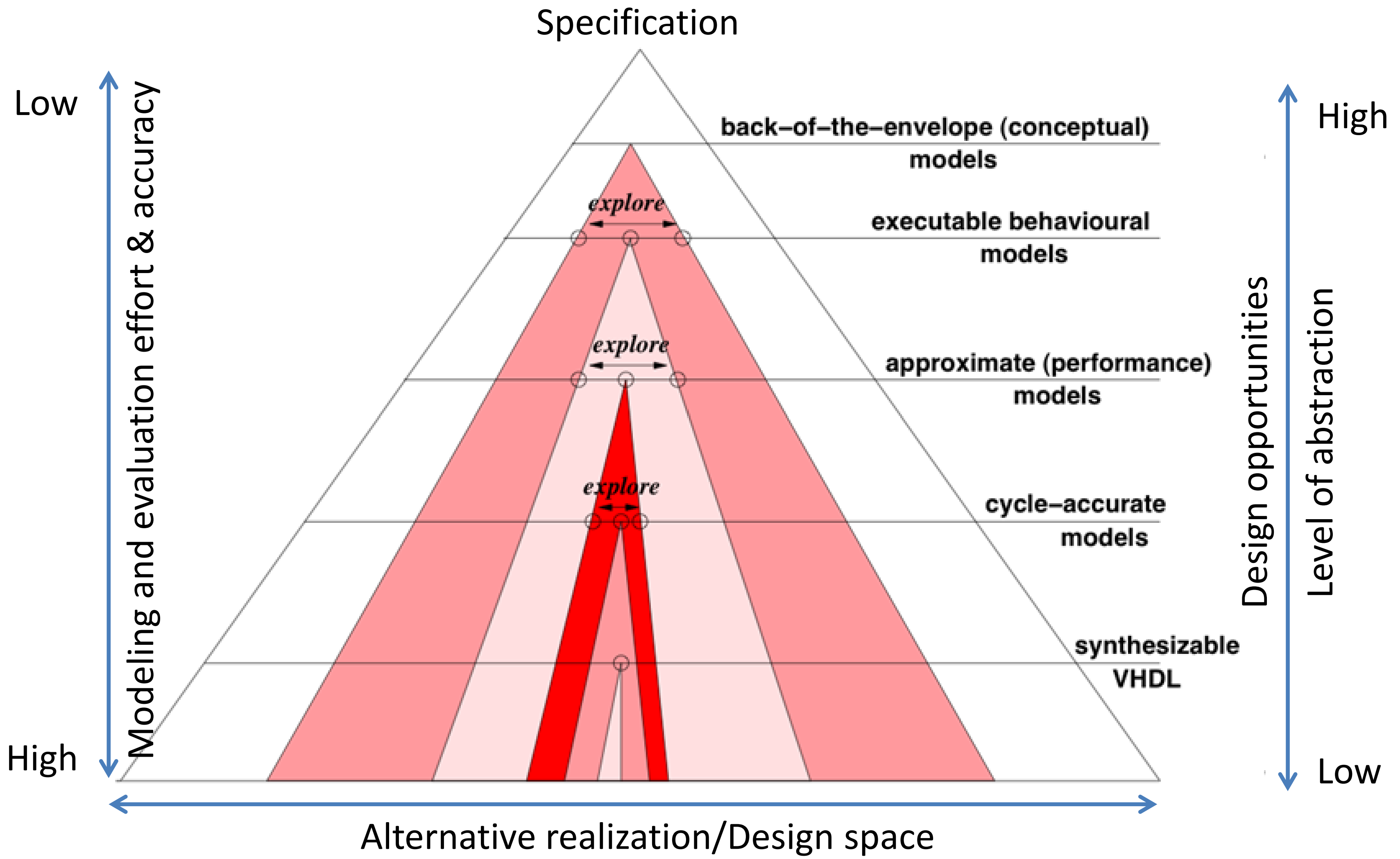


Specify and explore at different abstract levels

move down into lower abstraction levels

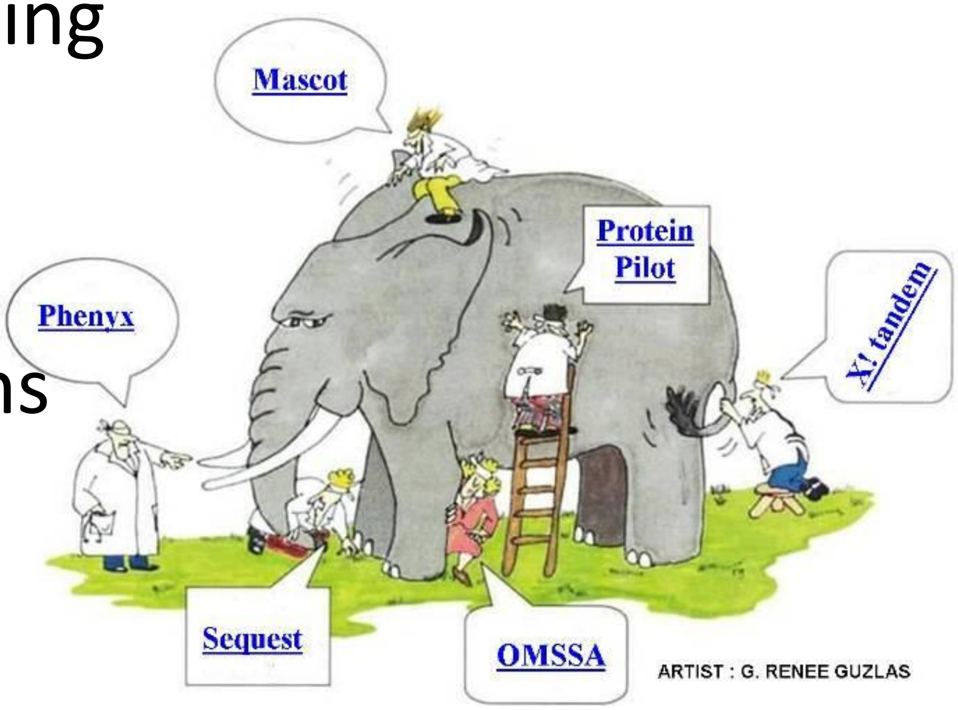
(keep the concept of separation of concerns)

Design-space exploration: Stepwise Refinement


Specification
Low
High
Modeling and evaluation effort & accuracy
back–of–the–envelope (conceptual) models
explore
executable behavioural models
explore
approximate (performance) models
explore
cycle–accurate models
synthesizable VHDL
Design opportunities
Level of abstraction
High
Low
Alternative realization/Design space

Search Algorithms

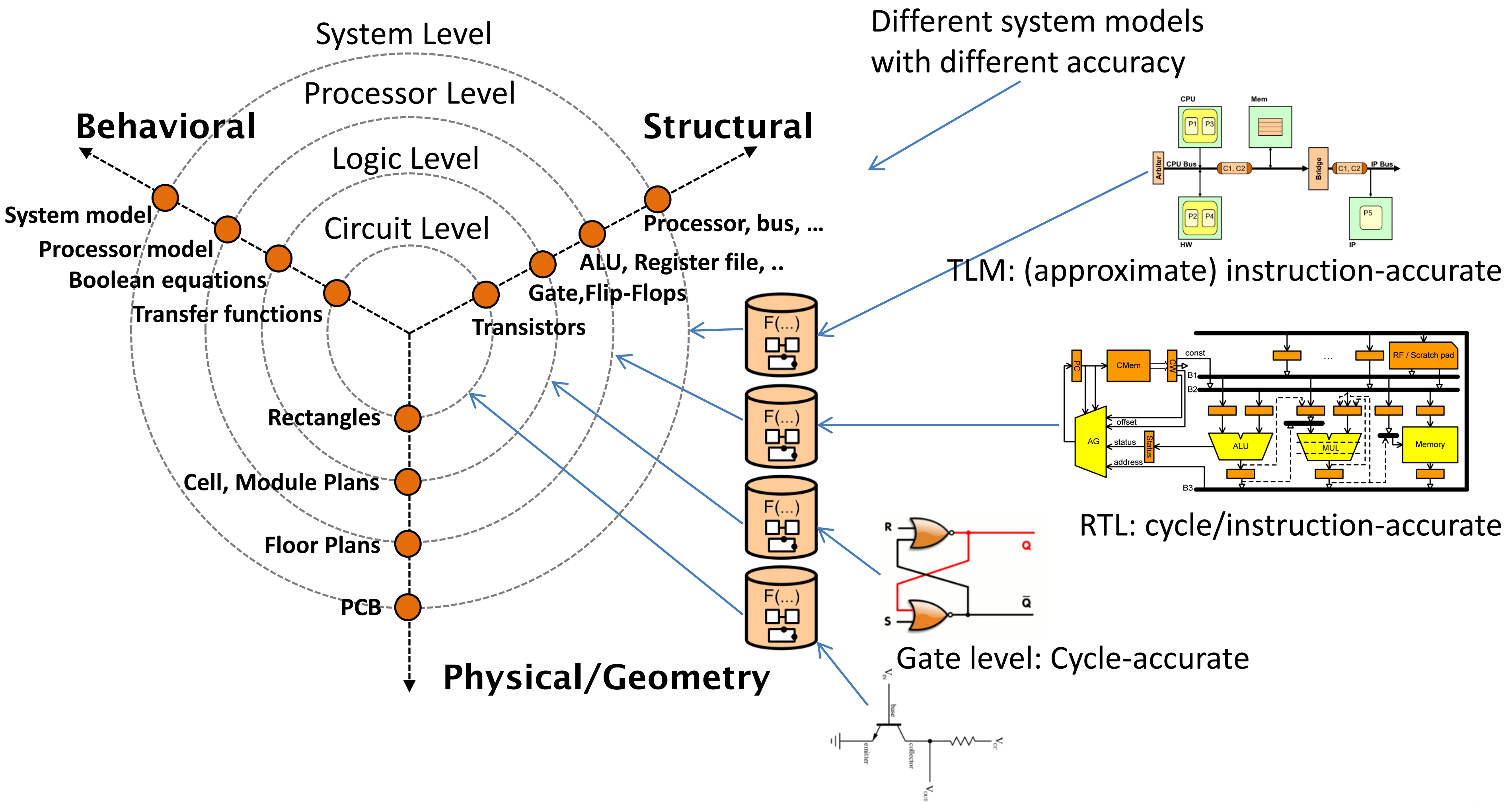
- Linear programming
- Dynamic programming
- Constraints programming
- Tabu search
- Simulated annealing
- Evolutionary algorithms



Summary (1)

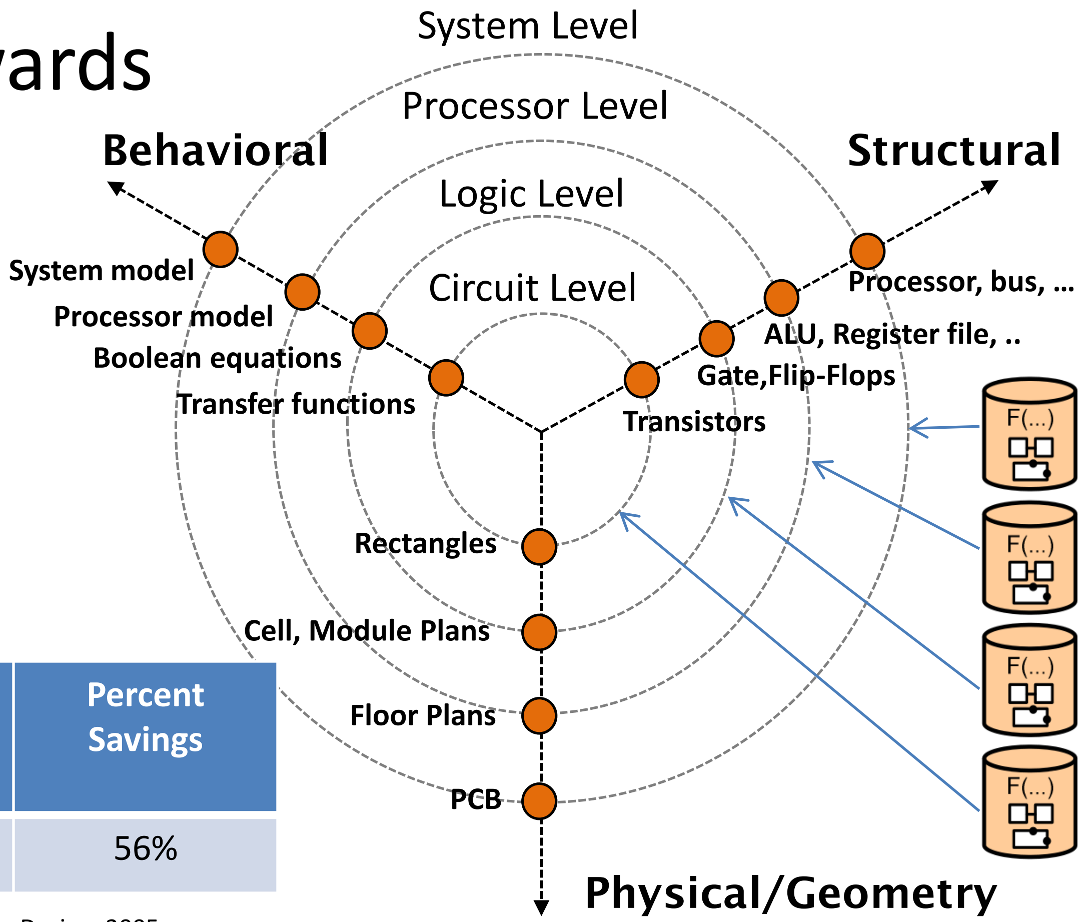
- Basic concepts of system design methodologies introduced
- Many different methodologies in use
  - One for every group, product, and company
- Methodologies differ in:
  - Input specification, MoC
  - Modeling styles and languages
  - Abstraction levels and amount of detail
  - Verification strategy and prototyping
  - CAD tools and component libraries
- Standards emerge slowly through experience

Summary (2)


System Level
Processor Level
Logic Level
Circuit Level
Behavioral
Structural
Physical/Geometry
System model
Processor model
Boolean equations
Transfer functions
Processor, bus, …
ALU, Register file, ..
Gate,Flip-Flops
Transistors
Rectangles
Cell, Module Plans
Floor Plans
PCB
F(...)
Different system models with different accuracy
CPU
Mem
P1
P3
Arbiter
CPU Bus
C1, C2
Bridge
IP Bus
P2
P4
P5
HW
IP
TLM: (approximate) instruction-accurate
PC
CMem
CW
const
RF / Scratch pad
B1
B2
offset
AG
status
Status
address
ALU
MUL
Memory
B3
RTL: cycle/instruction-accurate
R
S
Q
Q̄
Gate level: Cycle-accurate
V_IN
base
emitter
collector
V_CC
V_OUT

Conclusion

- Design moving towards system levels
- Design moving towards
  - model-based
  - platform-based
  - component-based



Average Spec. to RTL Cost: Before	Average Spec. to RTL Cost: After	Net Direct Savings	Percent Savings
$3.1M	$1.3M	$1.8M	56%

Source: Return on Investment in Simulink for Electronic Systems Design, 2005

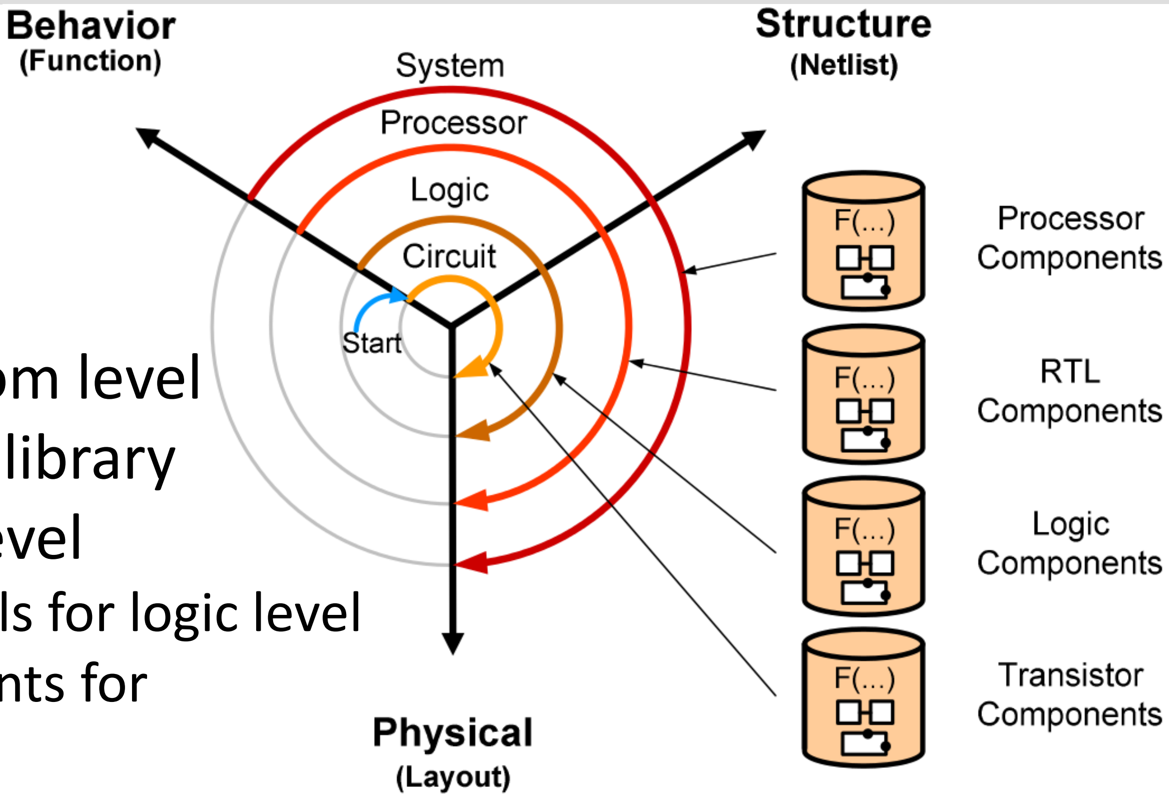
Below are some additional slides for references

Design methodologies

- Design methodology is a sequence of design models, components and tools used to design the product
- Methodologies evolve with technology, complexity, and automation
- A methodology depends on application, company and design group focus
- Standardization arrives when the cost of being special is too high

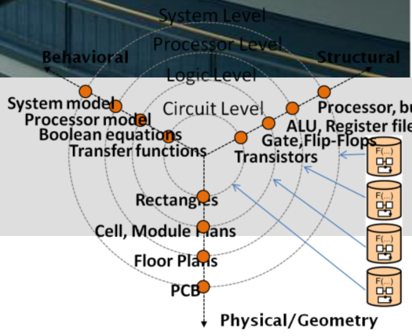
➢ Design Methodologies have been drastically changing with the increase in system complexity over the past half-century

Bottom-up Methodology



- Starts from the bottom level
- Each level generates library for the next higher level
  - Circuit: Standard cells for logic level
  - Logic: RTL components for processor level
  - Processor: Processing and communication components for system level
  - System: Embedded systems platforms for different applications
- Floorplaning and layout on each level

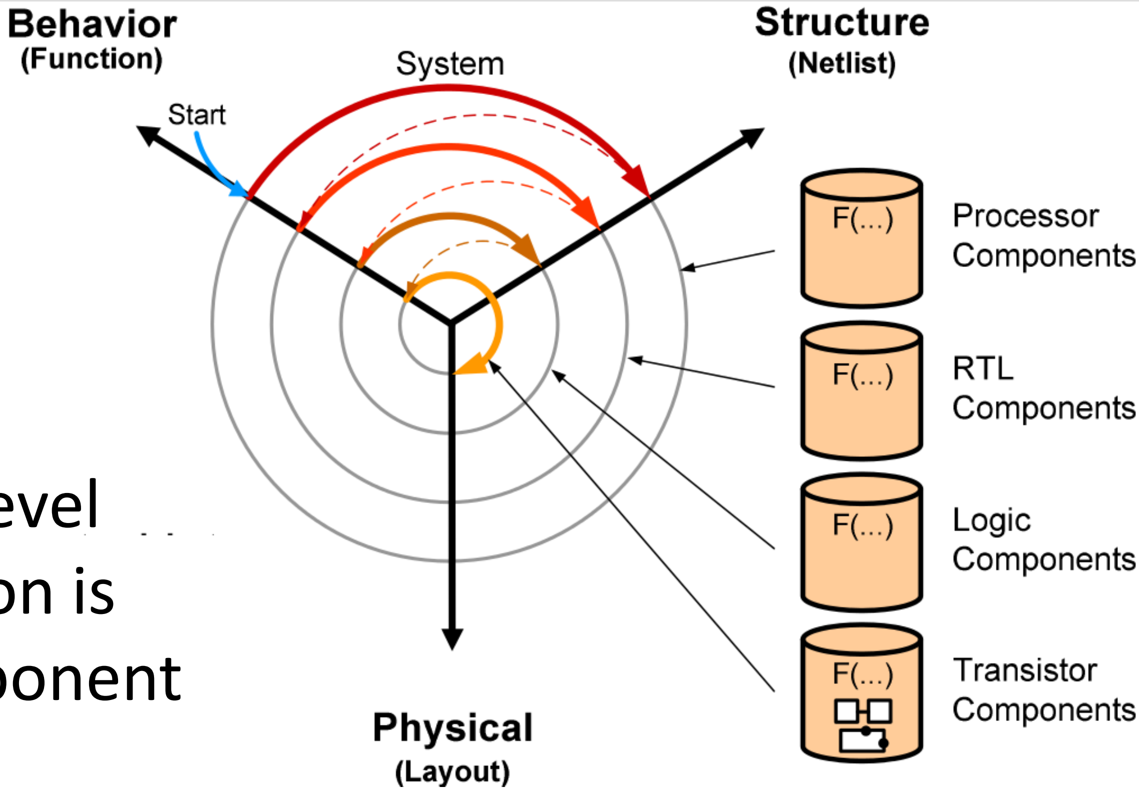
Bottom-up Methodology



- Pros
  - Abstraction levels clearly separated with its own library
  - Accurate metric estimation with layout on each level
  - Globally distributed development possible
  - Easy management

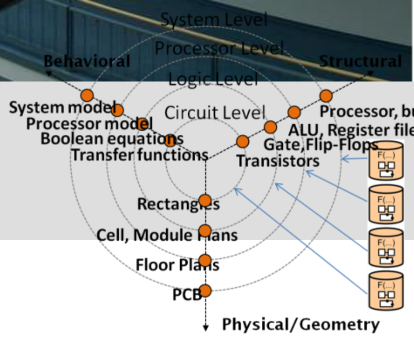
- Cons
  - An optimal library for each design is difficult to predict
    - All possible components with all possible parameters
    - All possible optimizations for all possible metrics
  - Library customization is outside the design group
  - Layout is performed on every level

Top-down Methodology



- Starts with the top level
- Functional description is converted into component netlist on each level
- Each component function is decomposed further on the next abstraction level
- Layout is given only for transistor components

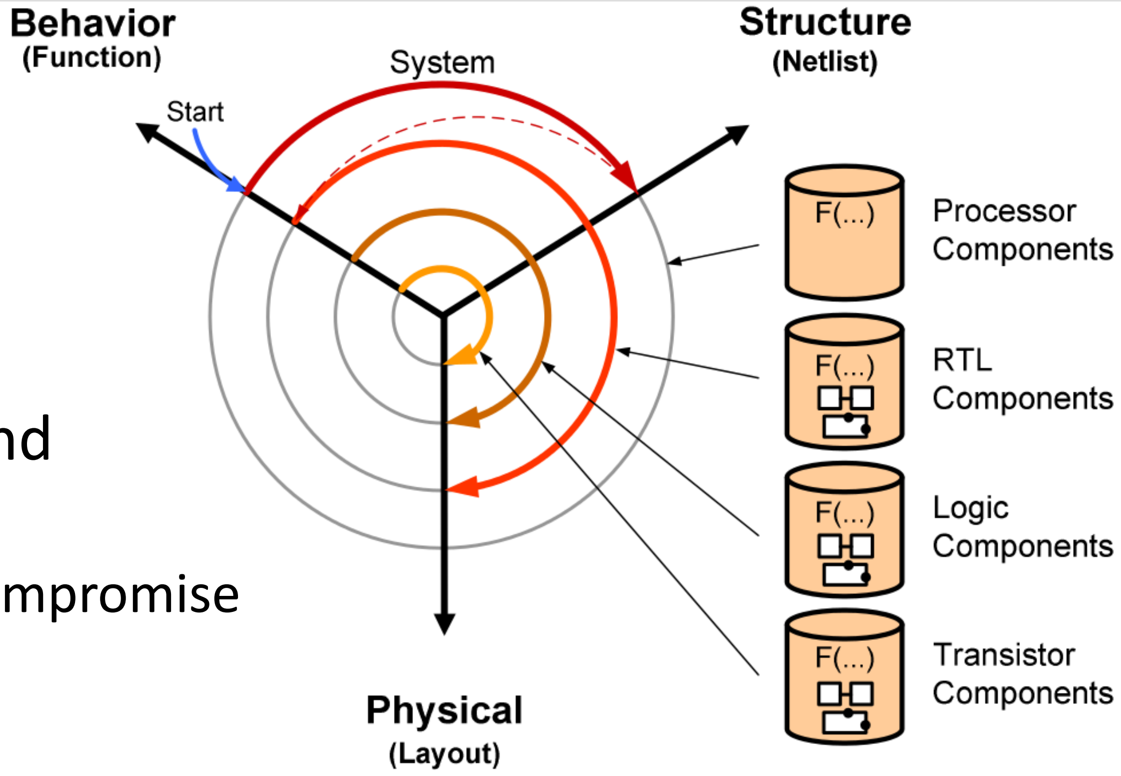
Top-down Methodology



- Pros
  - Highest level of customization possible on each abstraction level
  - Only one small transistor library needed
  - Only one layout design at the end

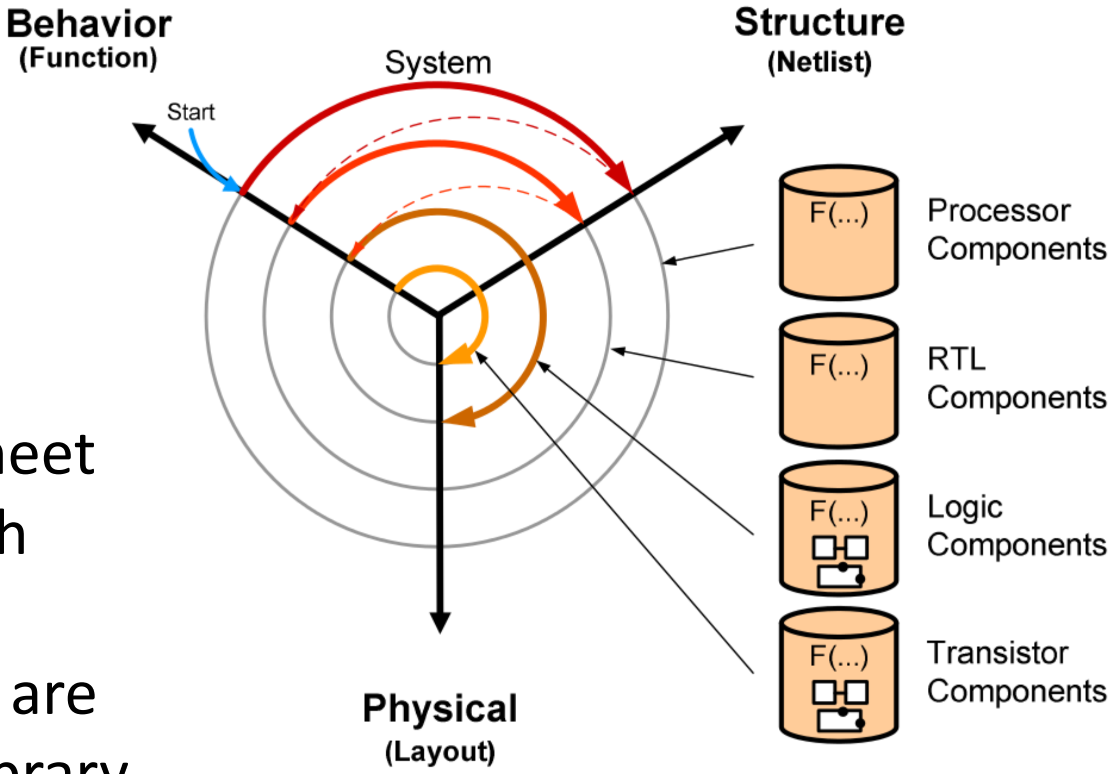
- Cons
  - Difficult metric estimation on upper levels since layout is not known until the end
  - Design decision impact on higher level not clear
  - Hot spot removal is difficult
  - Metric annotation (closure) from lower to higher levels needed during design iterations

Meet-in-the-Middle Methodology (Option 1)



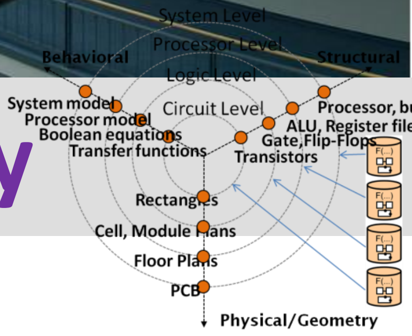
- Combines top-down and bottom-up
  - Synthesis vs. layout compromise
- Processor level is where they meet
- MoC is synthesized into processor components
- Processor components are synthesized with RTL library
- System layout is generated with RTL components

Meet-in-the-Middle Methodology (Option 2)



- RTL level where they meet
- MoC is synthesized with processor components
- Processor components are synthesized with RTL library
- RTL components are synthesized with standard cells
- System layout is performed with standard cells
- Two levels of layout

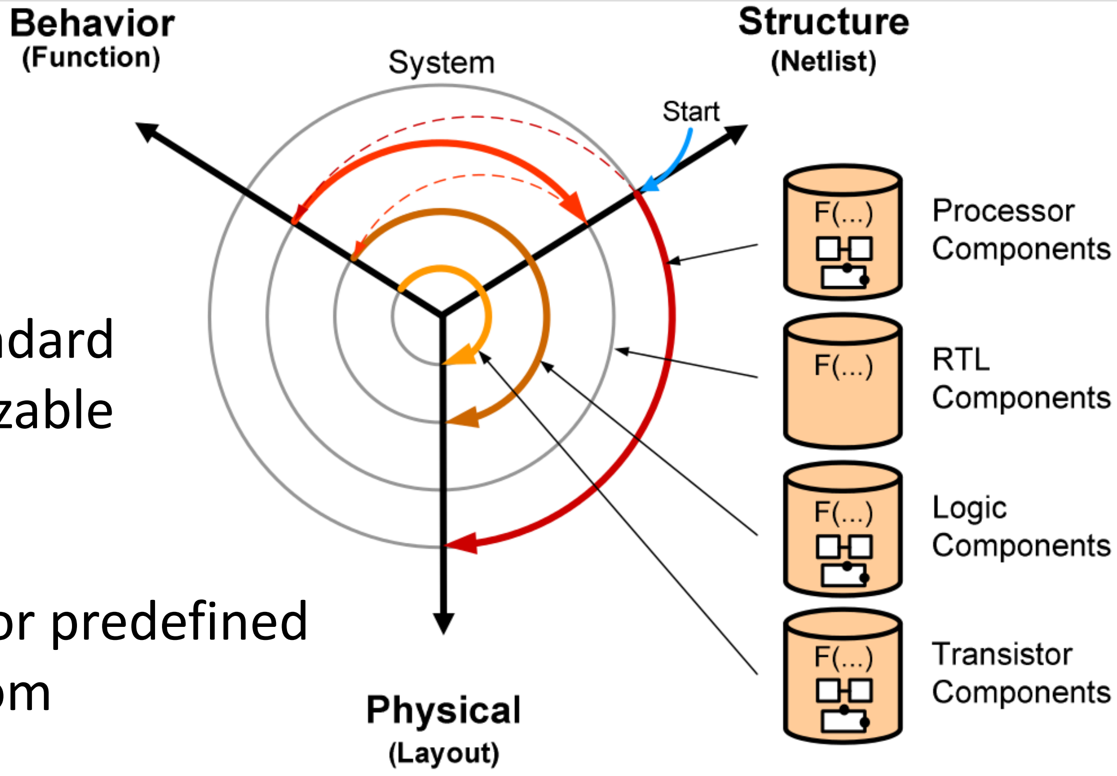
Meet-in-the-Middle Methodology



- Pros
  - Shorter synthesis
  - Less layout
  - Less libraries
  - Better metric closure

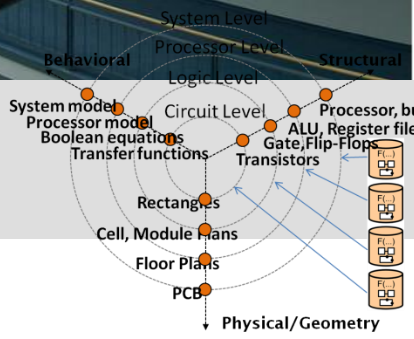
- Cons
  - Still needs libraries
  - More then one layout
  - Metric closure still needed
  - Library components may not be optimal

Platform Methodology



- System platform with standard components and synthesizable custom components for application optimization
- Layout is on system level or predefined with special area for custom components layout
- Custom components synthesized with RTL and logic and laid out with standard cells
- Custom components must fit into platform structure

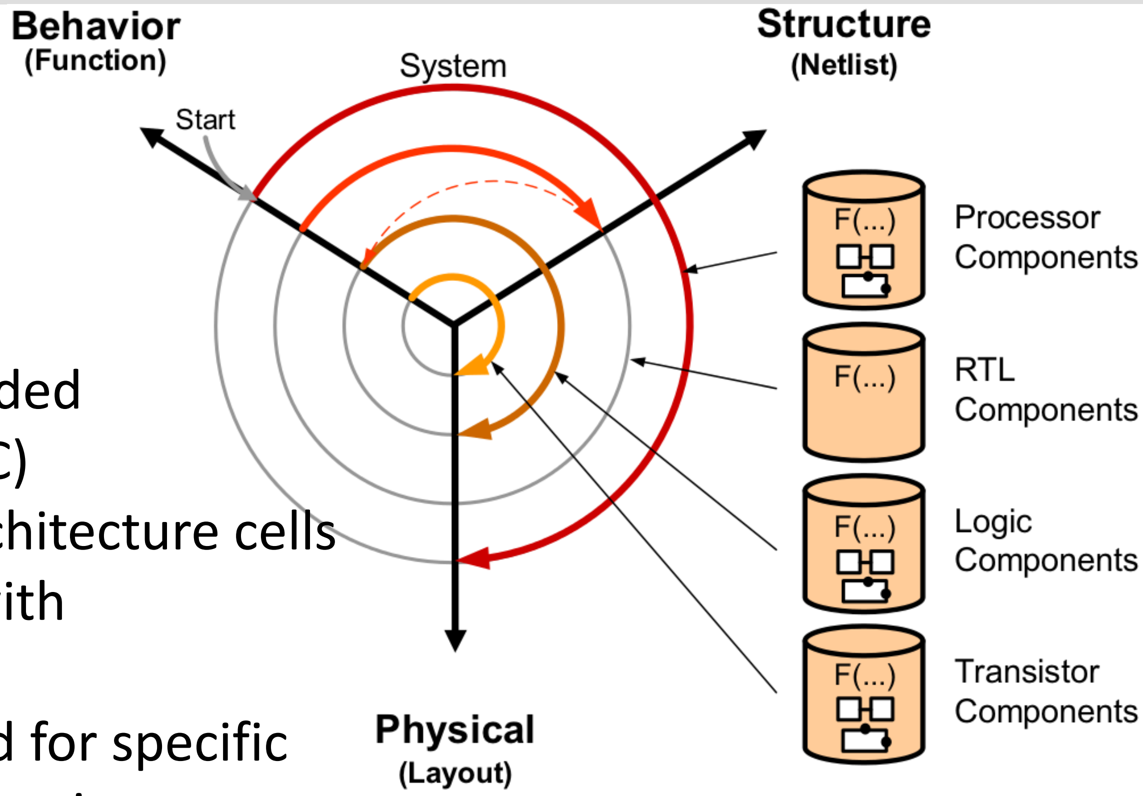
Platform Methodology



- Pros
  - Two types of layout: system layout for platform (could be predefined) and standard cell layout for custom components
  - Standard processors are available
  - Custom and interface components are added for optimization

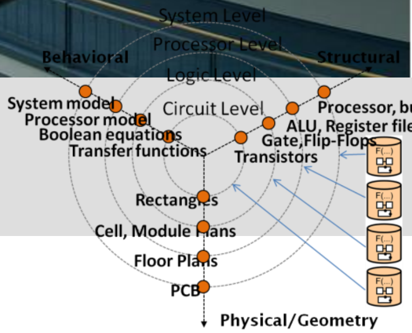
- Cons
  - Platform customization is still needed
  - SW and IF components synthesis required

System Methodology



- Methodology for embedded systems developers (ASIC)
- System platform with architecture cells
- Layout on system level with architecture cells
- Architecture cells defined for specific application and design metrics
- Architecture cells pre-synthesized with RTL and logic and laid out with standard cells
- A retargetable compiler for architecture cells

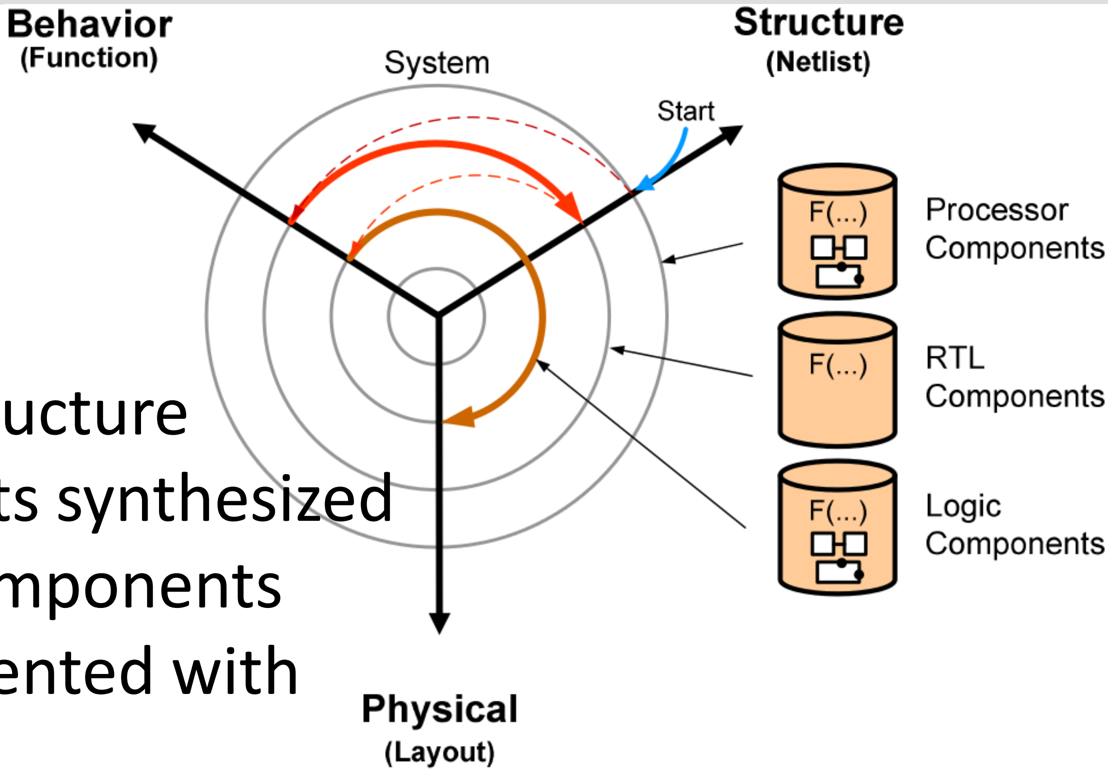
System Methodology



- Pros
  - Processor-level component only
  - Single retargetable compiler for all architecture cells
  - Processor-level layout
  - Methodology for application experts
  - Minimal knowledge of system and processor levels

- Cons
  - Architecture cell definition and library
  - IS definition
  - Change of mind

FPGA Methodology



- Starts with system structure
- Processor components synthesized with RTL and logic components
- Components implemented with LUT and BRAMs
- Layout only once
- Metric estimation very difficult
- Estimation is hidden in the FPGA supplier tools