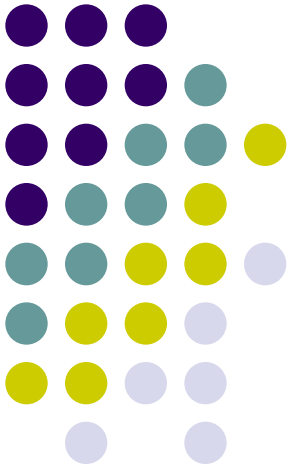
SEMICON Solutions

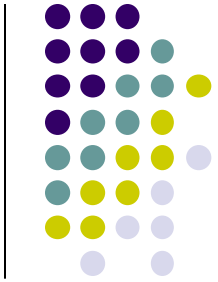
Principles of ASIC Design



Trình bày: Đặng Tường Dương

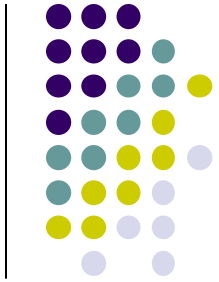
SEMICON
THINKING OUT OF THE BOX

Agenda



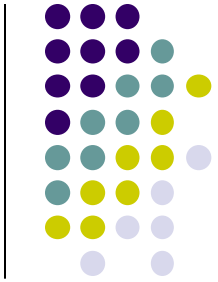
- Design Flow of ASICs
- ASICs & FPGAs
- CMOS Logic
- Design Specification
- RTL Coding
- Simulation
- Synthesis
- Floorplanning
- Placement
- Routing

Agenda



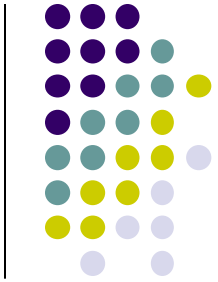
- Types of Design Styles
- ASIC & FPGA design flow
- Select ASIC or FPGA ???

What is ASIC?



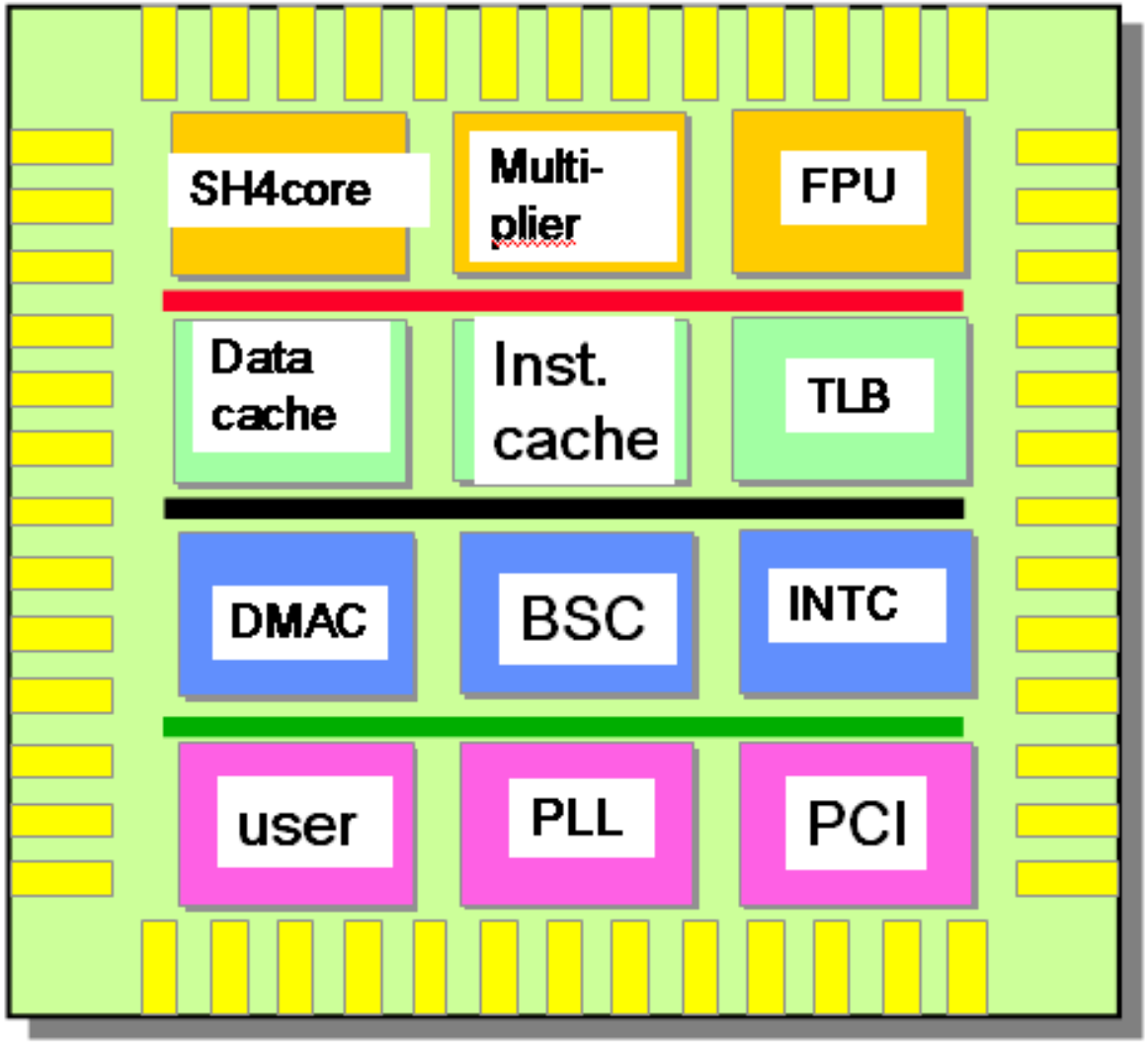
- ASIC (*application-specific integrated circuit*)
- ASIC is an IC designed for a specific application.
- ASIC is applied for microprocessor of Mobile phone, microprocessor of automatic machines, media, automobile, …

Types of ASICs

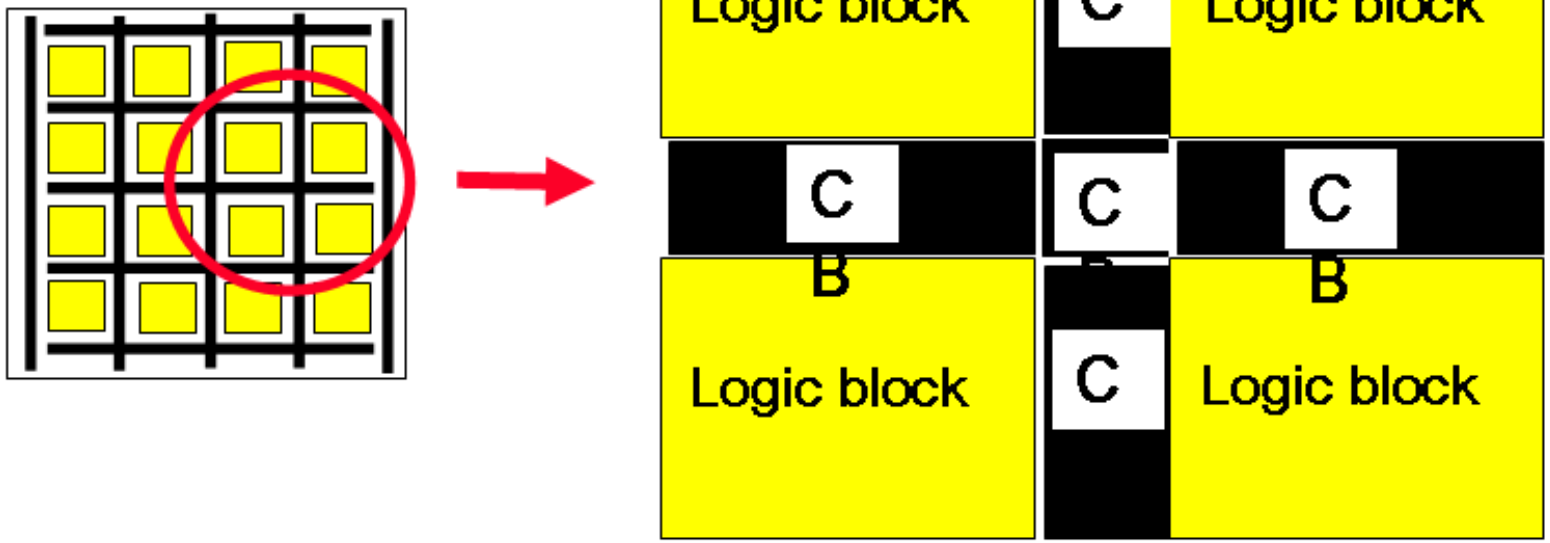


- Different types of ASICs are:
  - Full-custom ASICs
  - Semi-custom ASICs
    - Standard-cell–based ASICs (CBIC)
    - Gate-array–based ASICs
- Programmable ASICs
  - Programmable Logic Devices (PLD)
  - Field Programmable Gate Array (FPGA)

ASIC Architecture



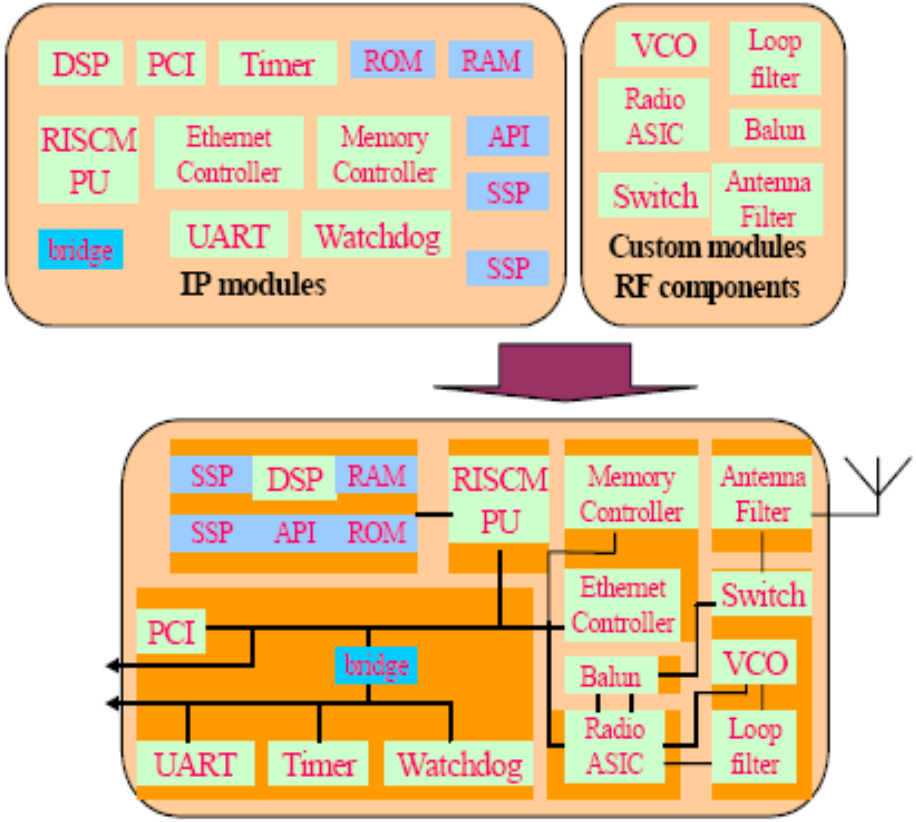
FPGA Architecture

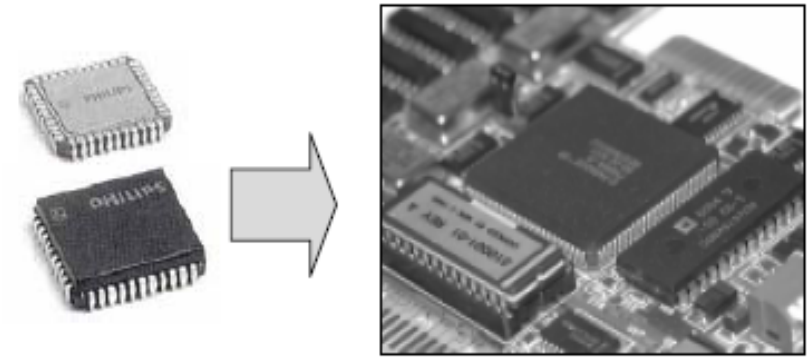


CB:Connection Block

FPGA:Field Programmable Gate Array

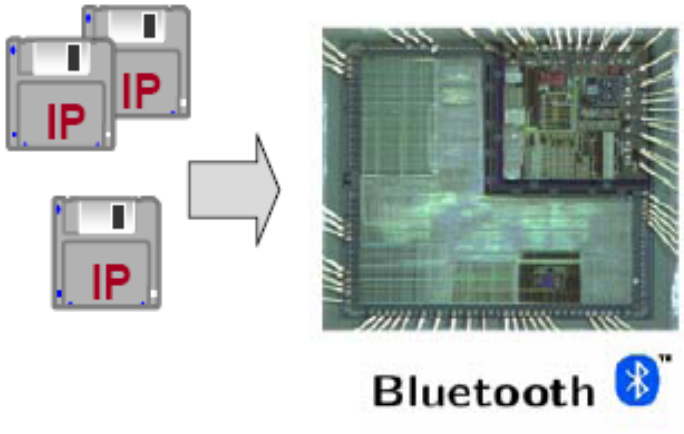
An example of ASIC/FPGA Design





Yesterday/Today: Real Components

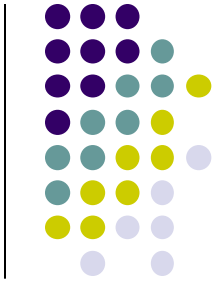
IP: Intellectual Property



Today/Future: Virtual Components

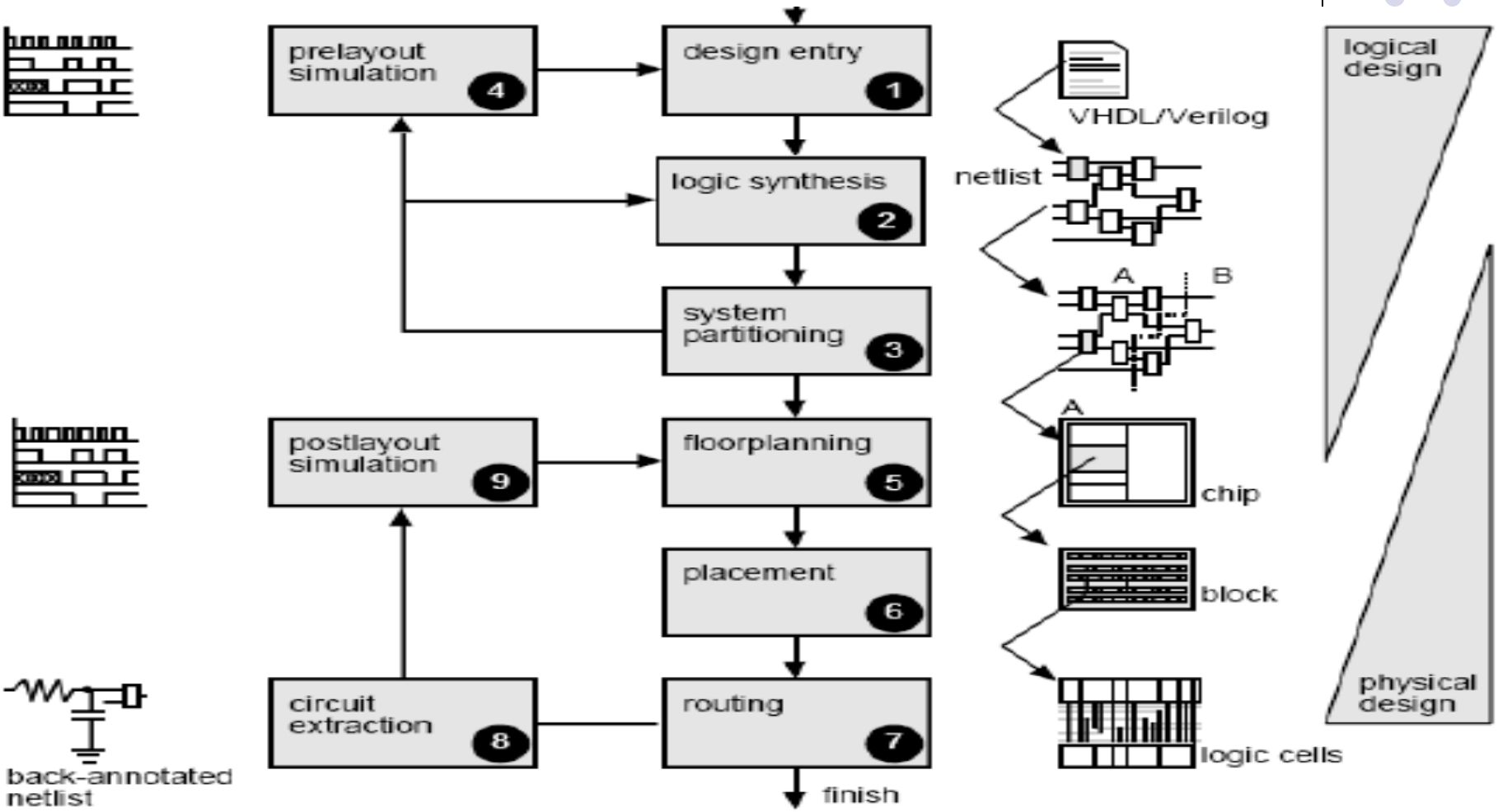
SEMICON

ASIC Design Flow

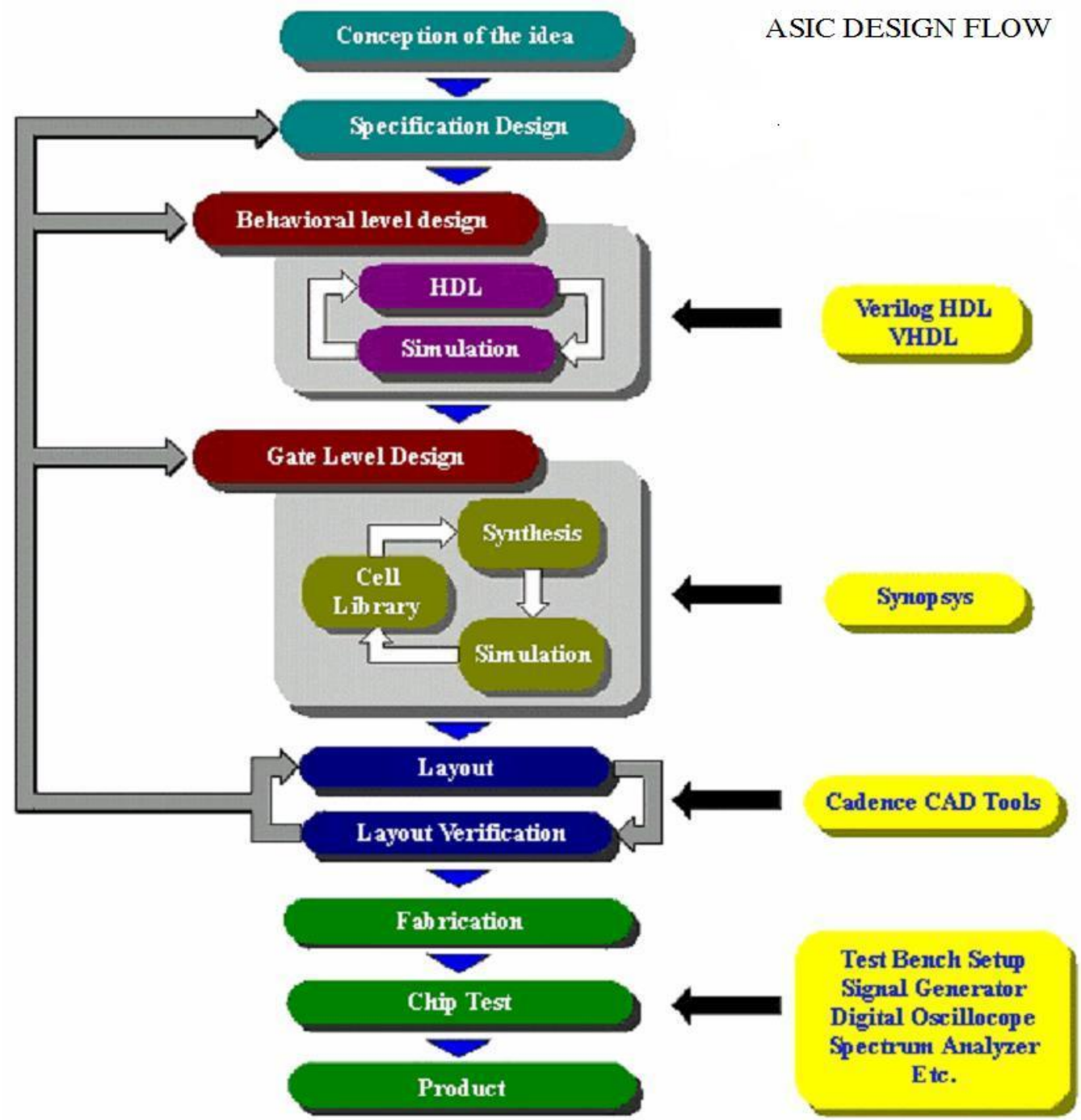


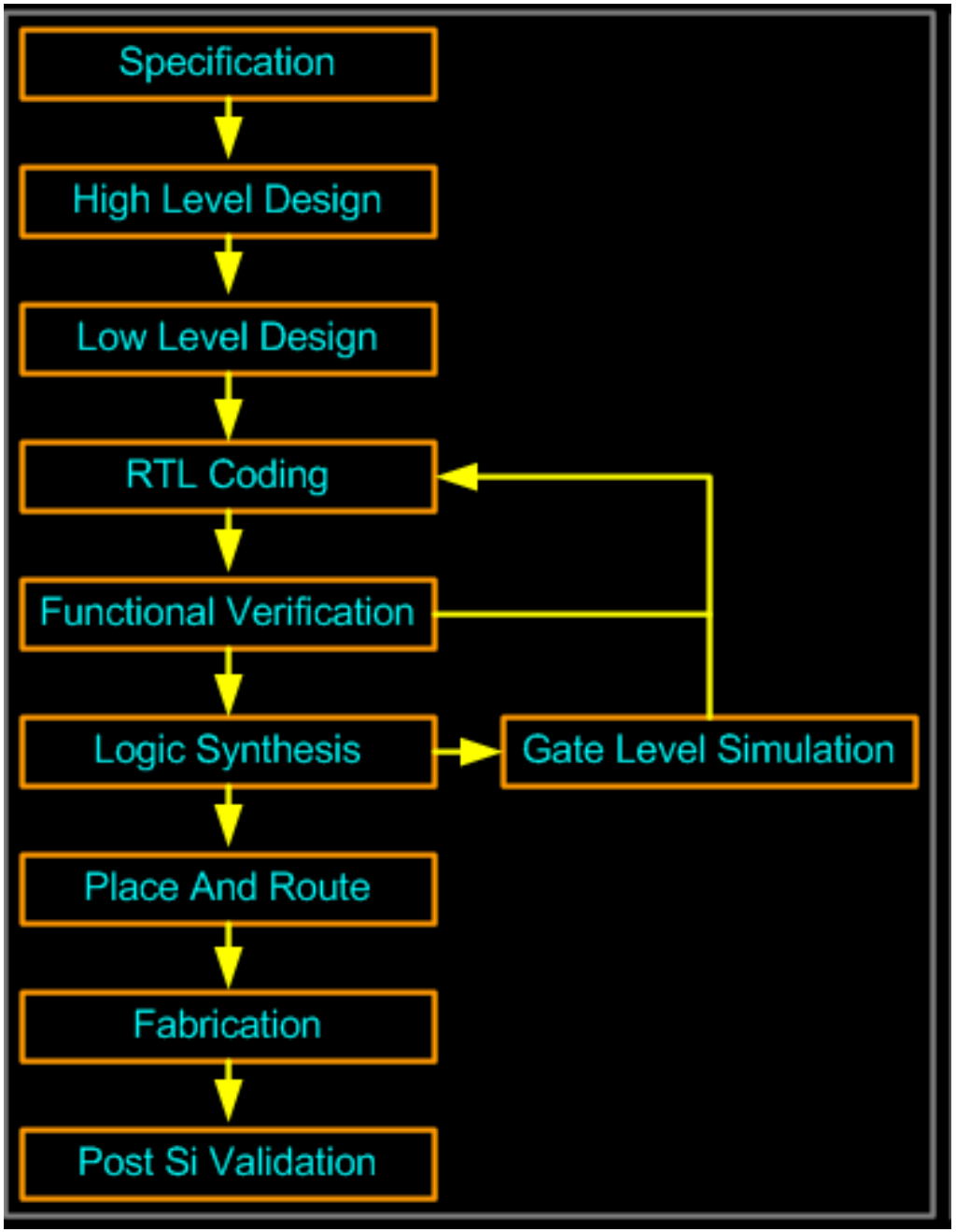
- A design flow is a sequence of steps to design an ASIC
  - **Design entry:** Using a hardware description language (HDL) or schematic entry.
  - **Logic synthesis:** Produces a netlist-logic cells and their connections.
  - **System partitioning:** Divide a large system into ASIC-sized pieces.
  - **Prelayout simulation:** Check to see if the design functions correctly.
  - **Floorplanning:** Arrange the blocks of the netlist on the chip.
  - **Placement:** Decide the locations of cells in a block.
  - **Routing:** Make the connections between cells and blocks.
  - **Extraction:** Determine the resistance and capacitance of the interconnect.
  - **Postlayout simulation:** Check to see the design still works with the added loads of the interconnect.

ASIC Design Flow

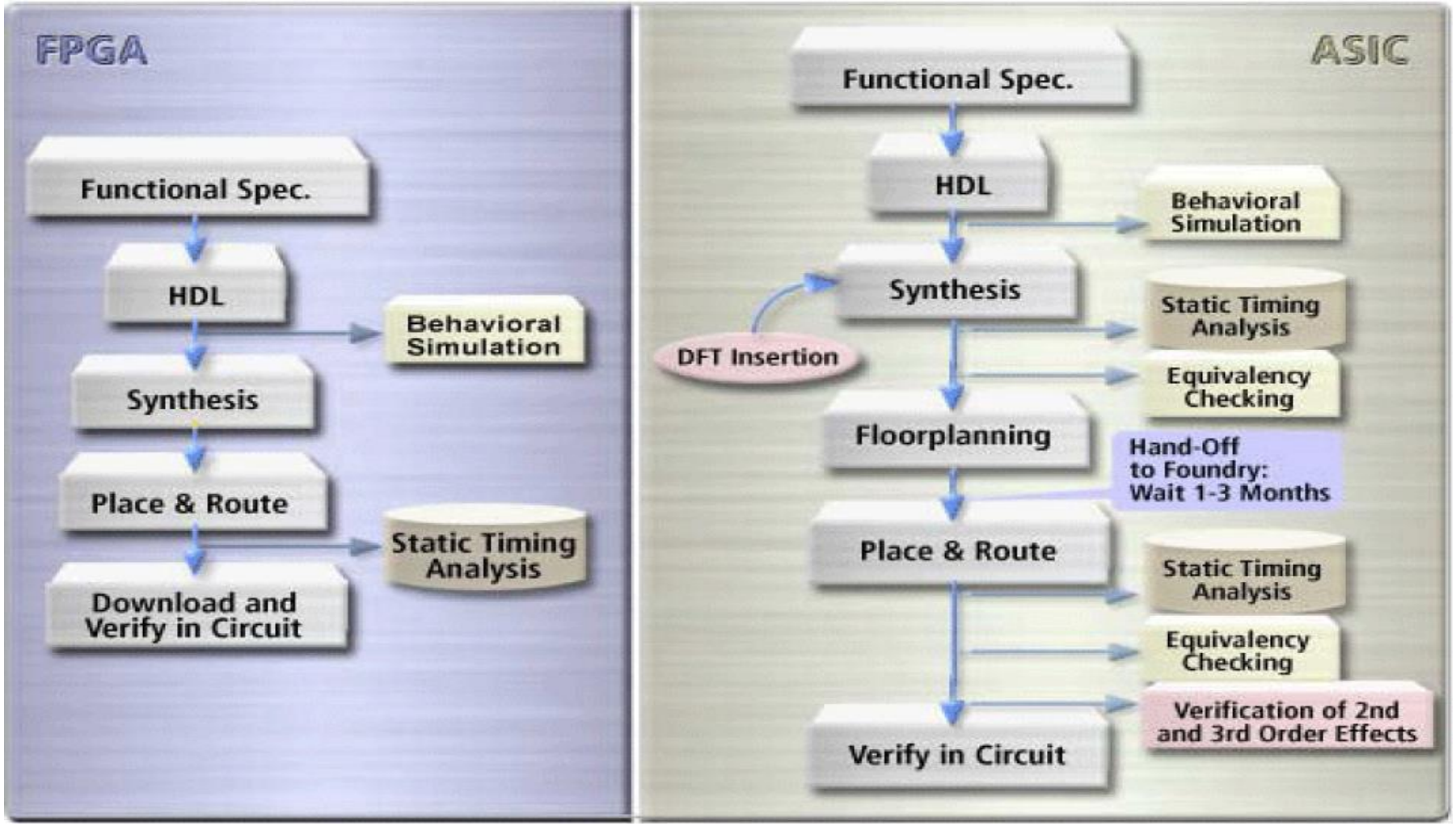


1-4: Logic Design 5-9: Physical Design

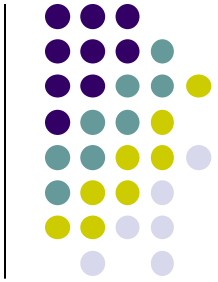

ASIC DESIGN FLOW
Conception of the idea
Specification Design
Behavioral level design
HDL
Simulation
Verilog HDL
VHDL
Gate Level Design
Synthesis
Cell
Library
Simulation
Synopsys
Layout
Layout Verification
Cadence CAD Tools
Fabrication
Chip Test
Product
Test Bench Setup
Signal Generator
Digital Oscillocope
Spectrum Analyzer
Etc.


Specification
High Level Design
Low Level Design
RTL Coding
Functional Verification
Logic Synthesis
Gate Level Simulation
Place And Route
Fabrication
Post Si Validation

FPGA & ASIC
Design Flow Comparison

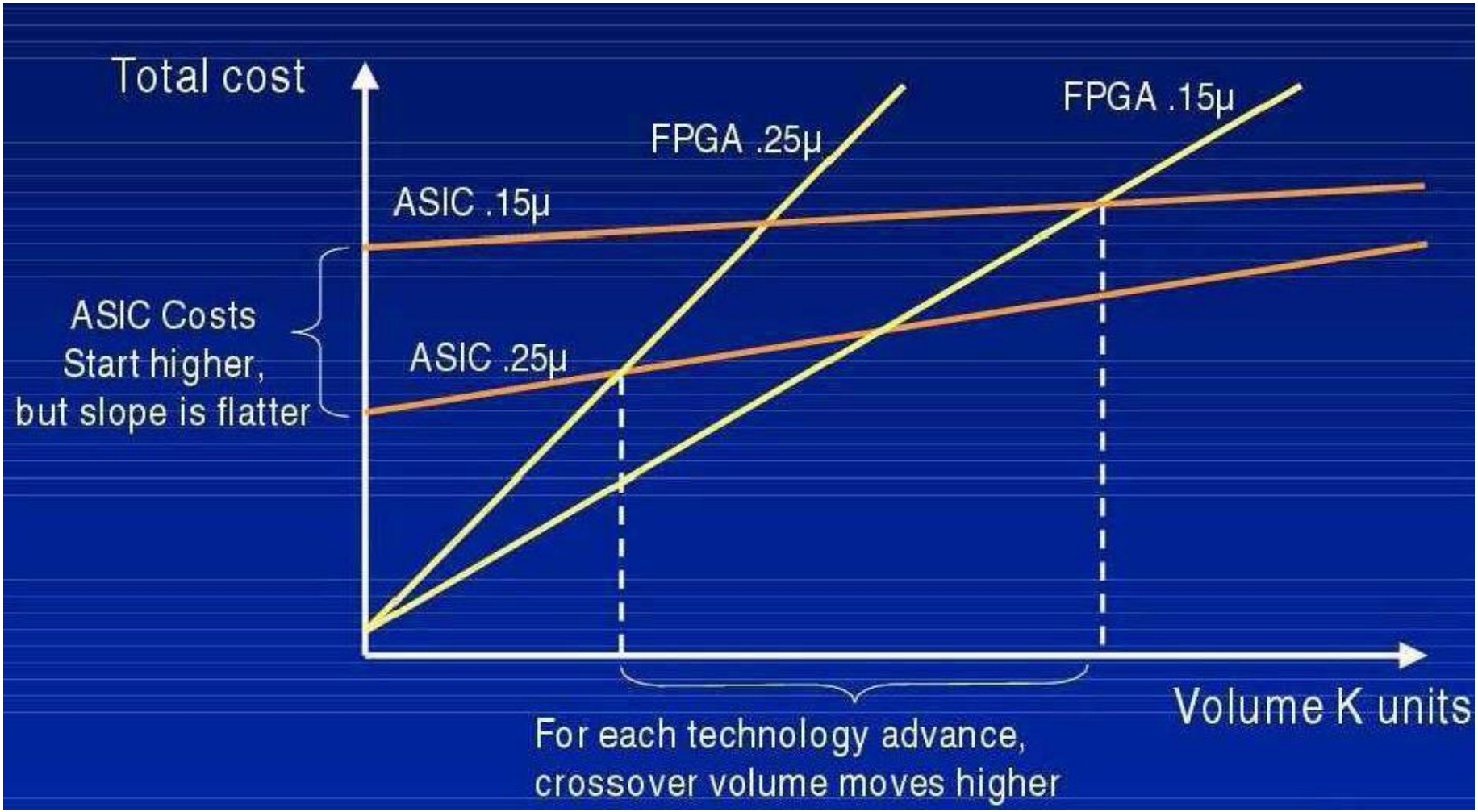


FPGA & ASIC Advantages



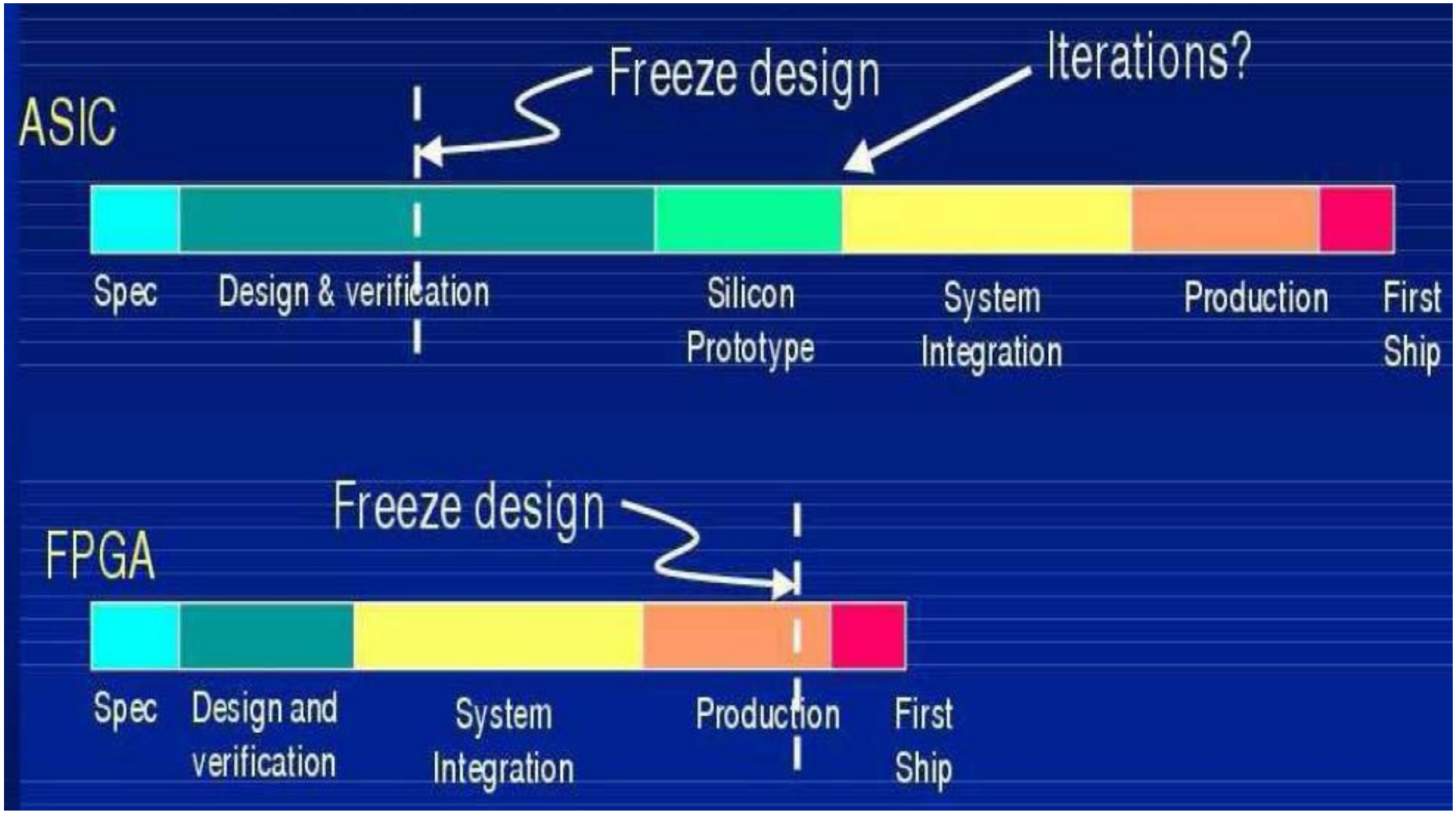
FPGA Design Advantages	ASIC Design Advantages
Faster time-to-market - no layout, masks or other manufacturing steps are needed	**Full custom capability** - for design since device is manufactured to design specs
No upfront NRE (non recurring expenses) - costs typically associated with an ASIC design	**Lower unit costs** - for very high volume designs
Simpler design cycle - due to software that handles much of the routing, placement, and timing	**Smaller form factor** - since device is manufactured to design specs
More predictable project cycle - due to elimination of potential re-spins, wafer capacities, etc.	**Higher raw internal clock speeds**
Field reprogramability - a new bitstream can be uploaded remotely	

Unit Cost Analysis

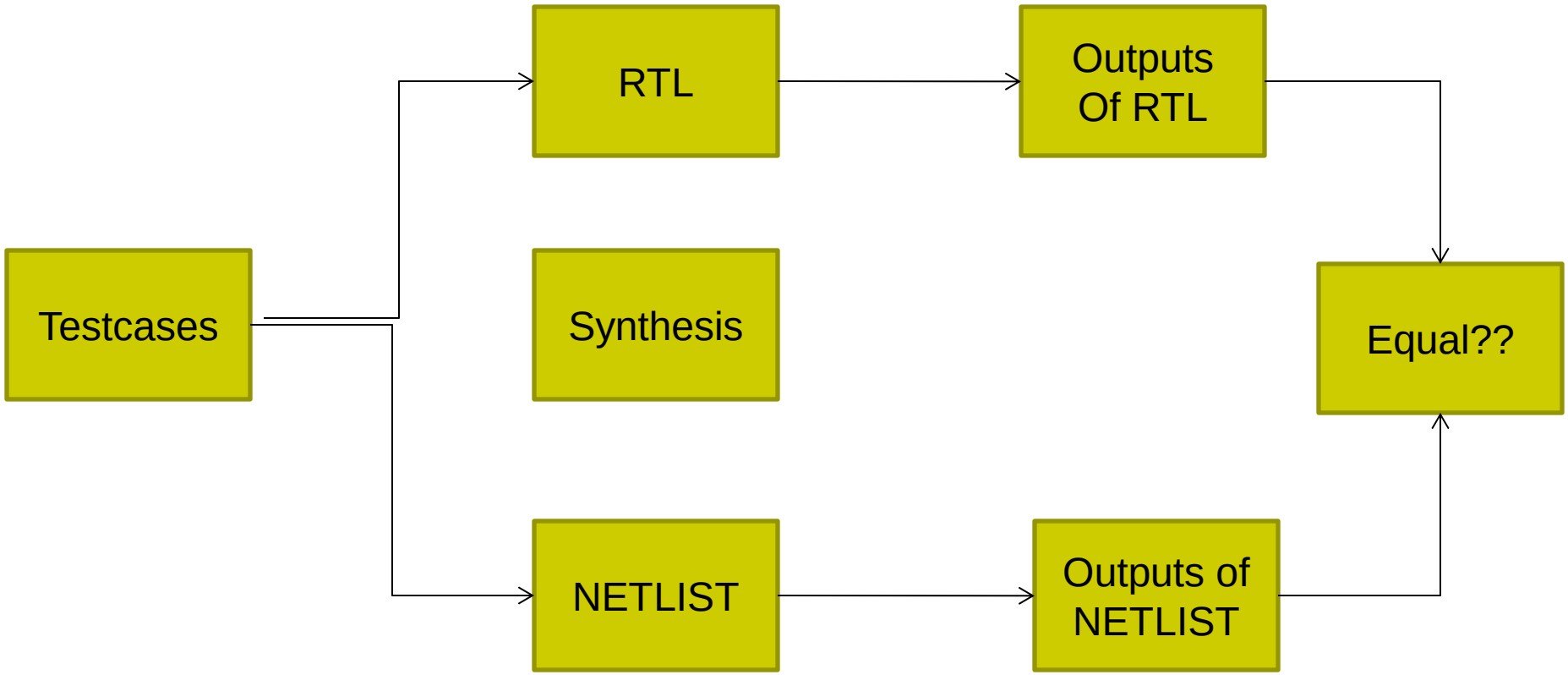


Time to Market

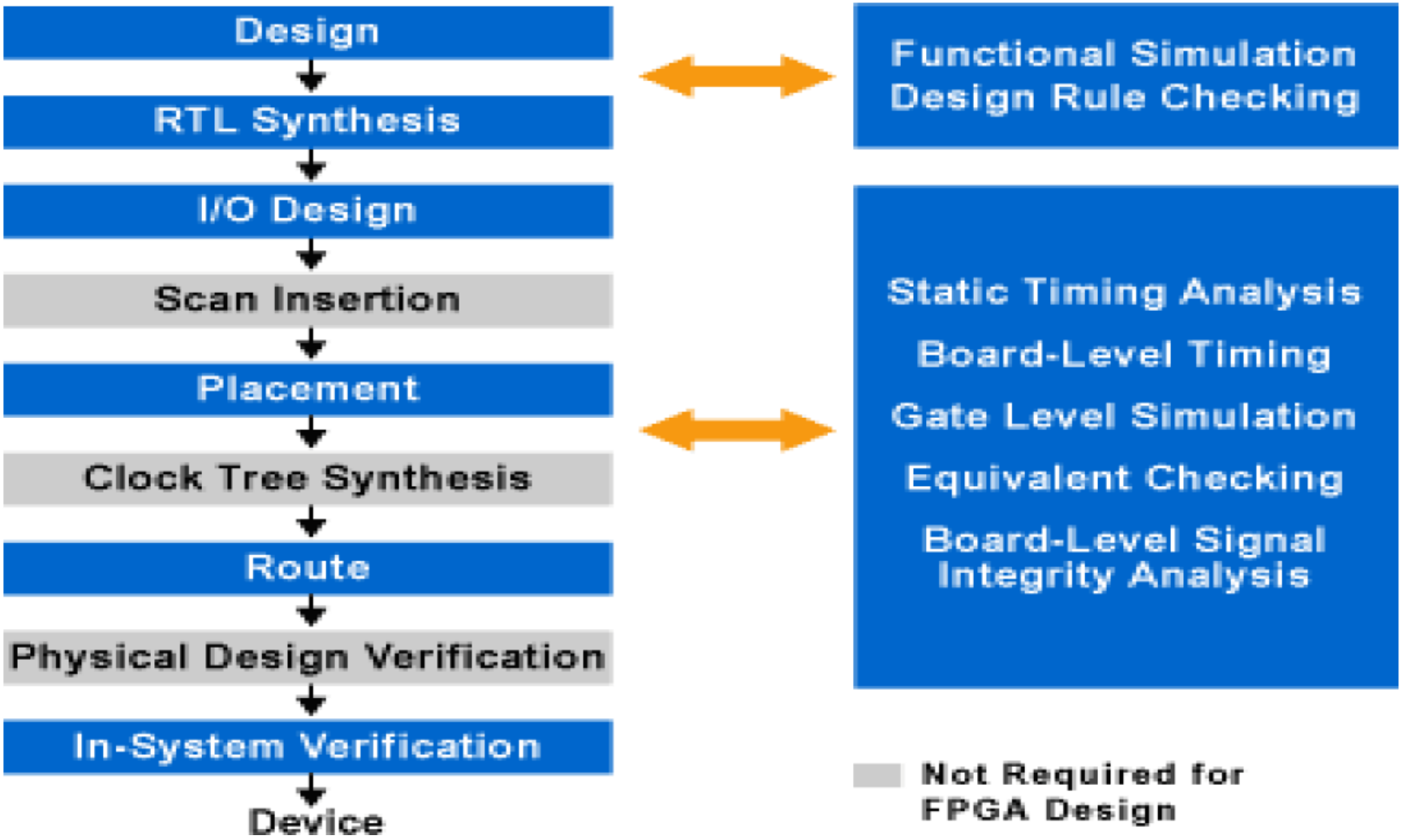
Design Cycle

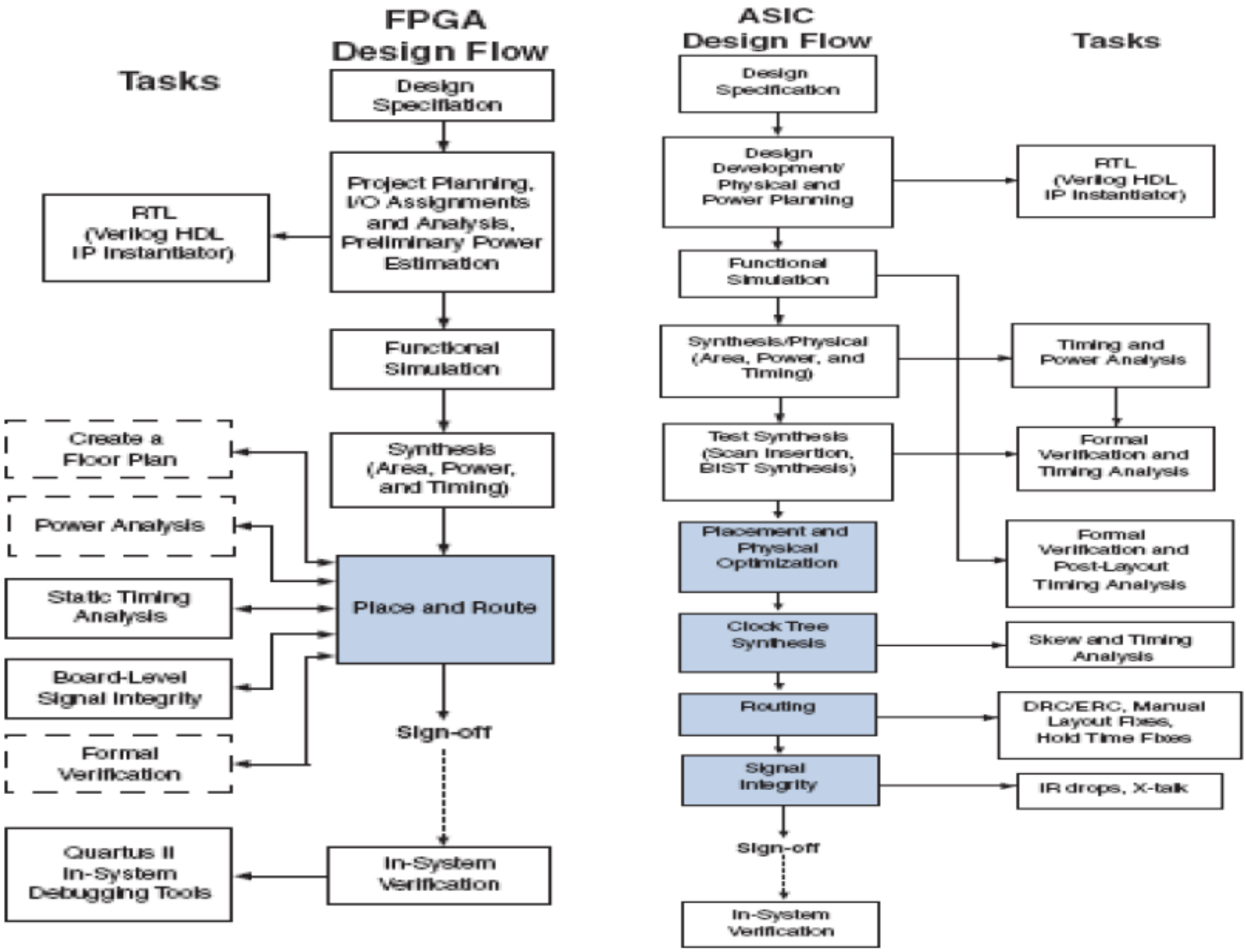


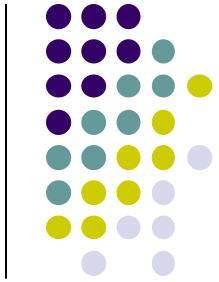
Equivalent Checking Idea



FPGA and ASIC Design Flows Fundamentally Similar



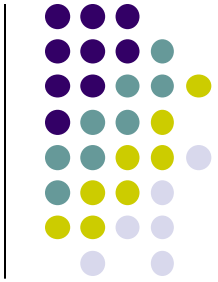

FPGA
Design Flow
Tasks
Design Specification
Project Planning, I/O Assignments and Analysis, Preliminary Power Estimation
RTL (Verilog HDL IP Instantiator)
Functional Simulation
Synthesis (Area, Power, and Timing)
Create a Floor Plan
Power Analysis
Static Timing Analysis
Board-Level Signal Integrity
Formal Verification
Place and Route
Sign-off
In-System Verification
Quartus II In-System Debugging Tools
ASIC
Design Flow
Tasks
Design Specification
Design Development/ Physical and Power Planning
RTL (Verilog HDL IP Instantiator)
Functional Simulation
Synthesis/Physical (Area, Power, and Timing)
Timing and Power Analysis
Test Synthesis (Scan Insertion, BIST Synthesis)
Formal Verification and Timing Analysis
Placement and Physical Optimization
Formal Verification and Post-Layout Timing Analysis
Clock Tree Synthesis
Skew and Timing Analysis
Routing
DRC/ERC, Manual Layout Fixes, Hold Time Fixes
Signal Integrity
IR drops, X-talk
Sign-off
In-System Verification



CMOS Logic

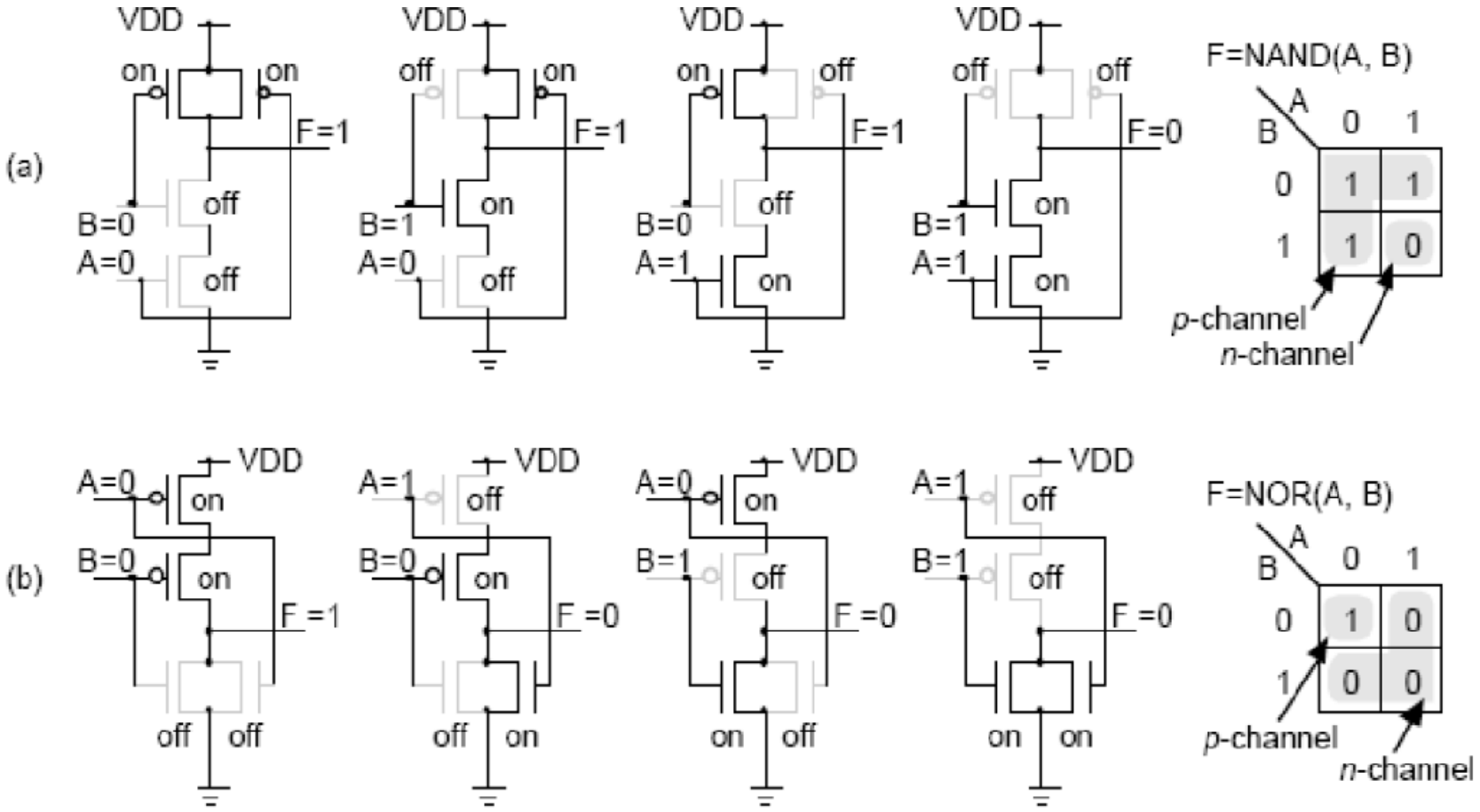
(Complementary Metal-Oxide Semiconductor logic)

Agenda



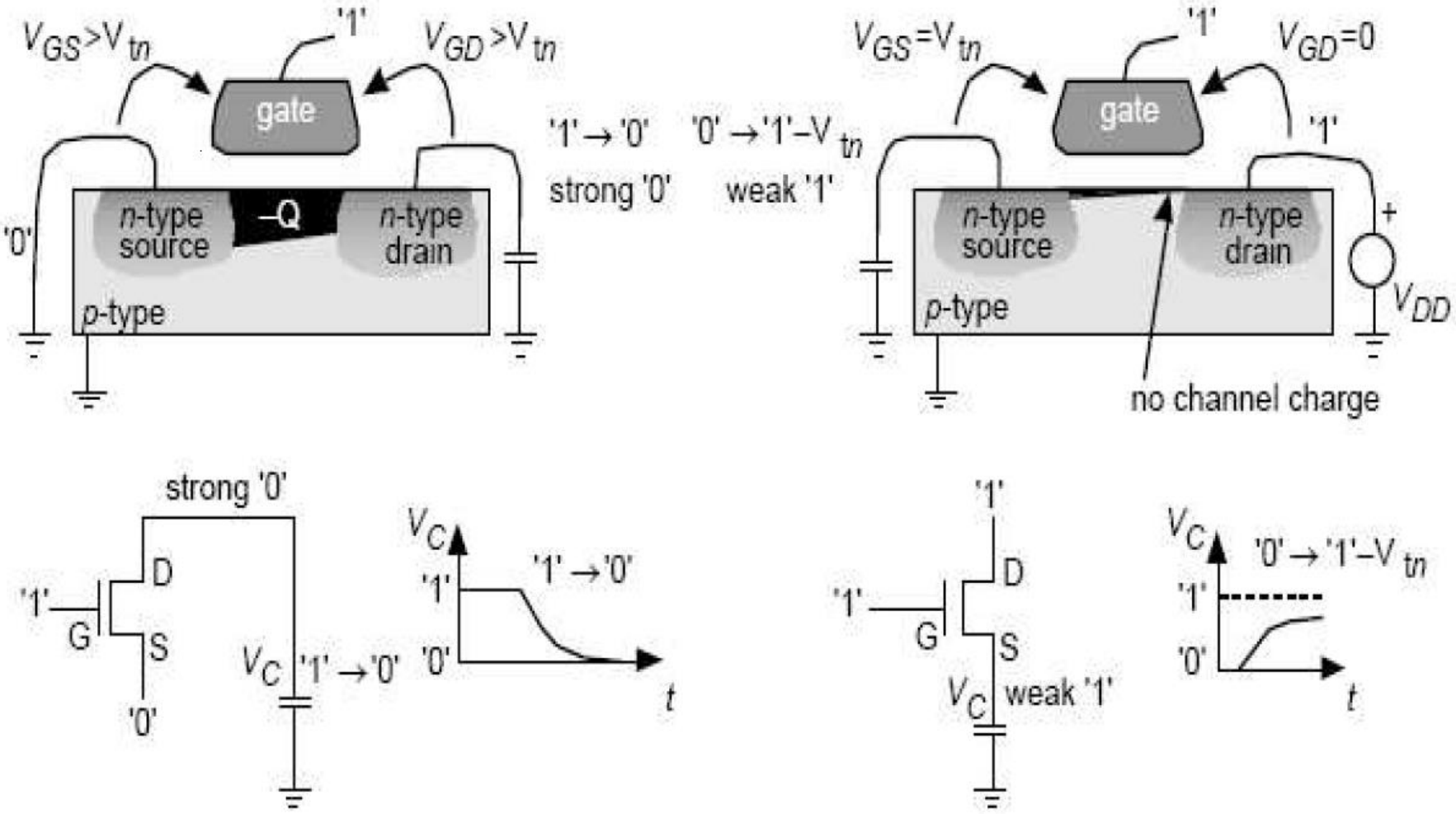
- CMOS logic
- Logic levels
- Transmission Gates
- Sequential Logic Cells
- Datapath Logic Cells

CMOS Logic

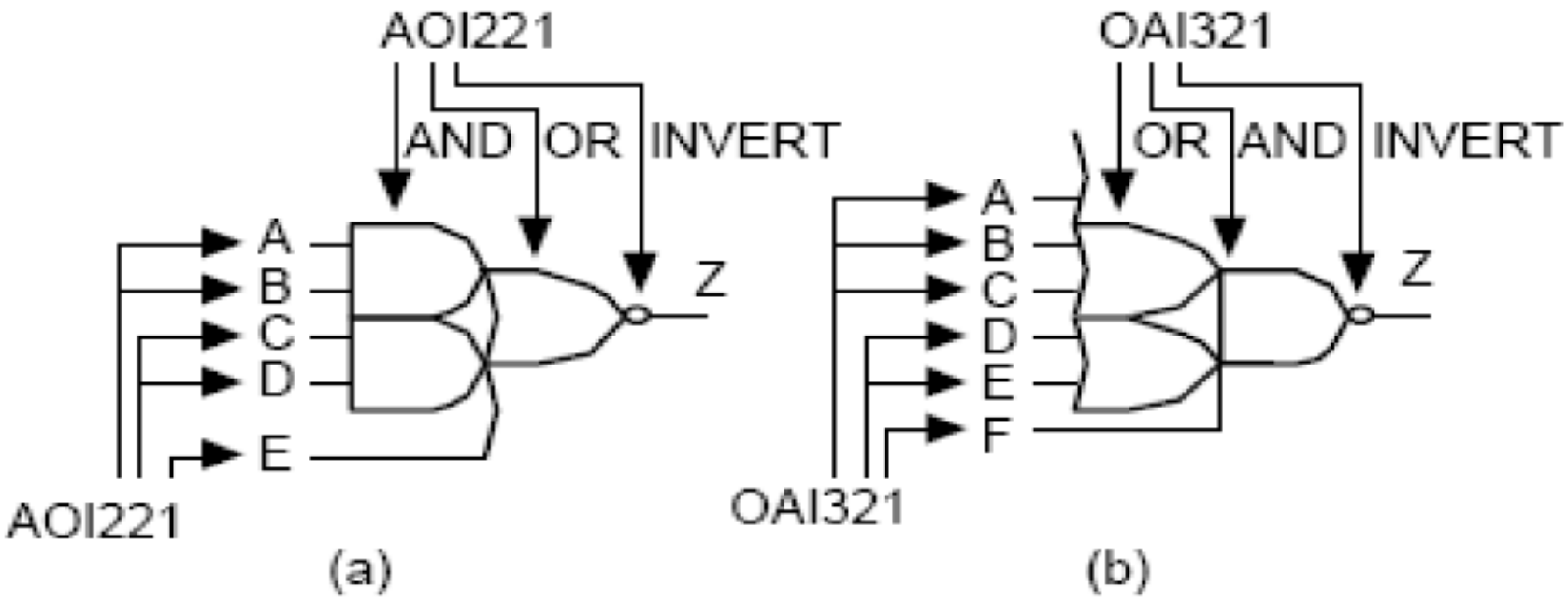


CMOS logic • a two-input **NAND gate** • a two-input **NOR gate** • **Good '1's** • **Good '0's**

Logic Levels



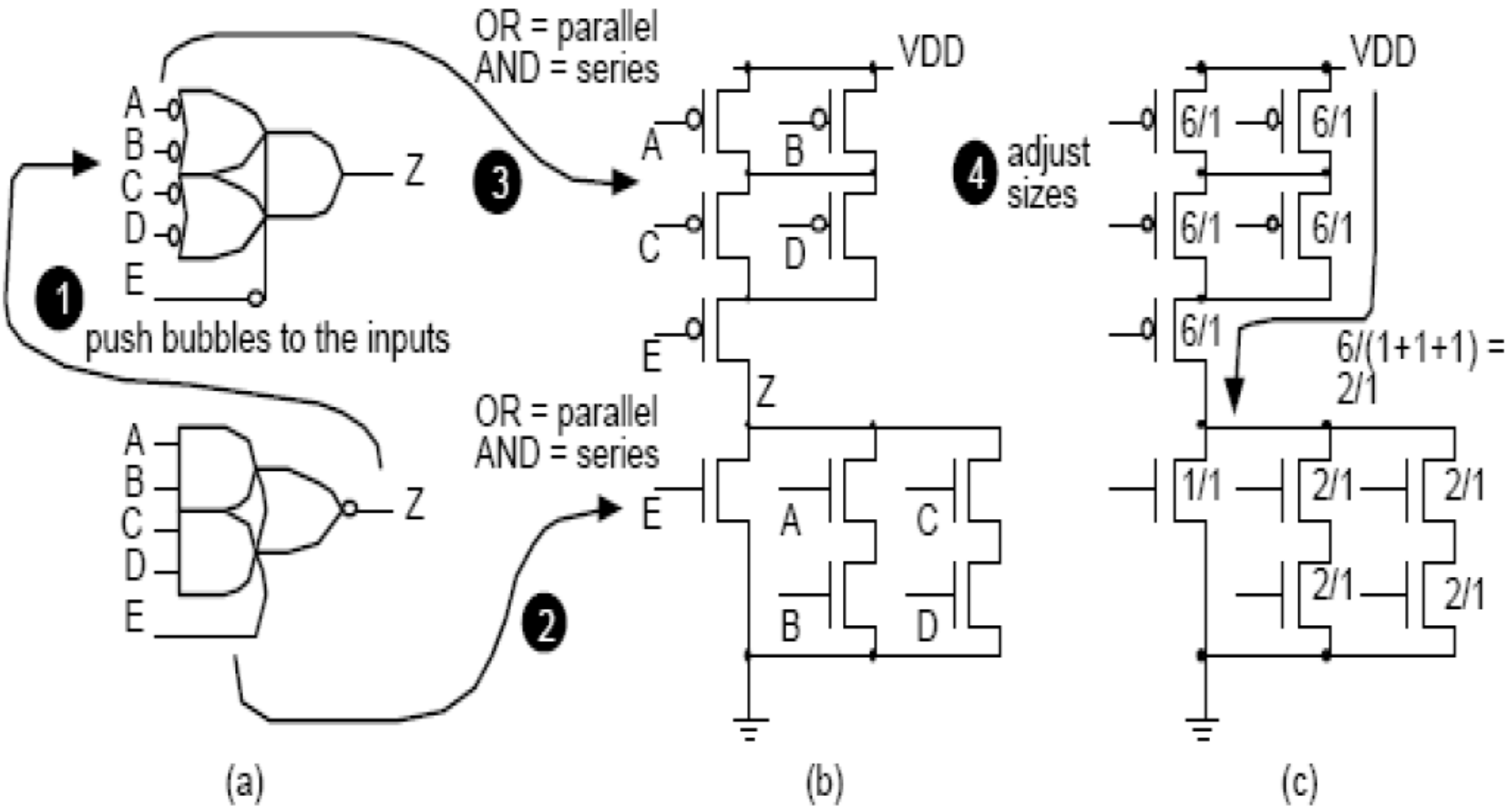
Naming of complex CMOS combinational logic cells



Drive Strength

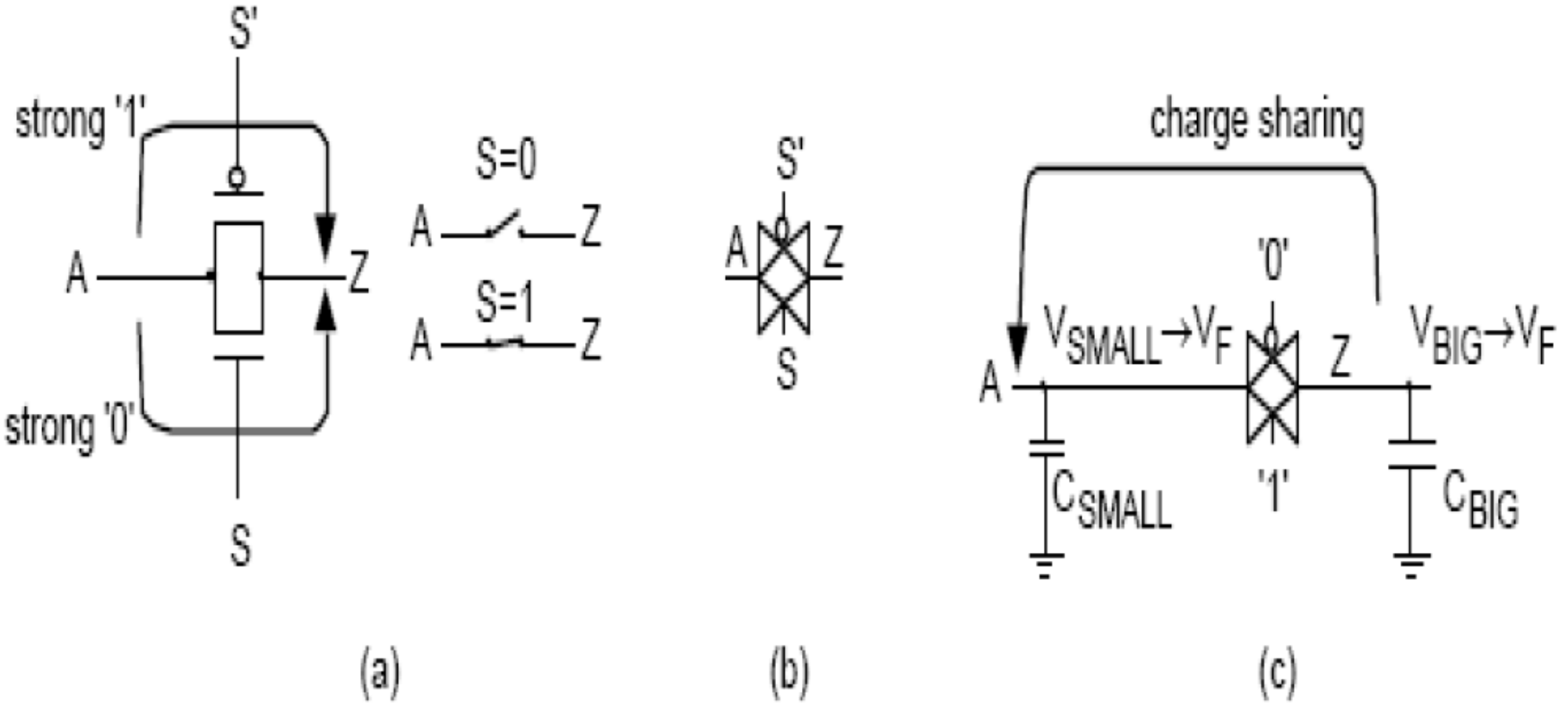
We **ratio** a cell to adjust its **drive strength** and make b*n*=b*p* to create equal rise and fall times

Naming of complex CMOS combinational logic cells

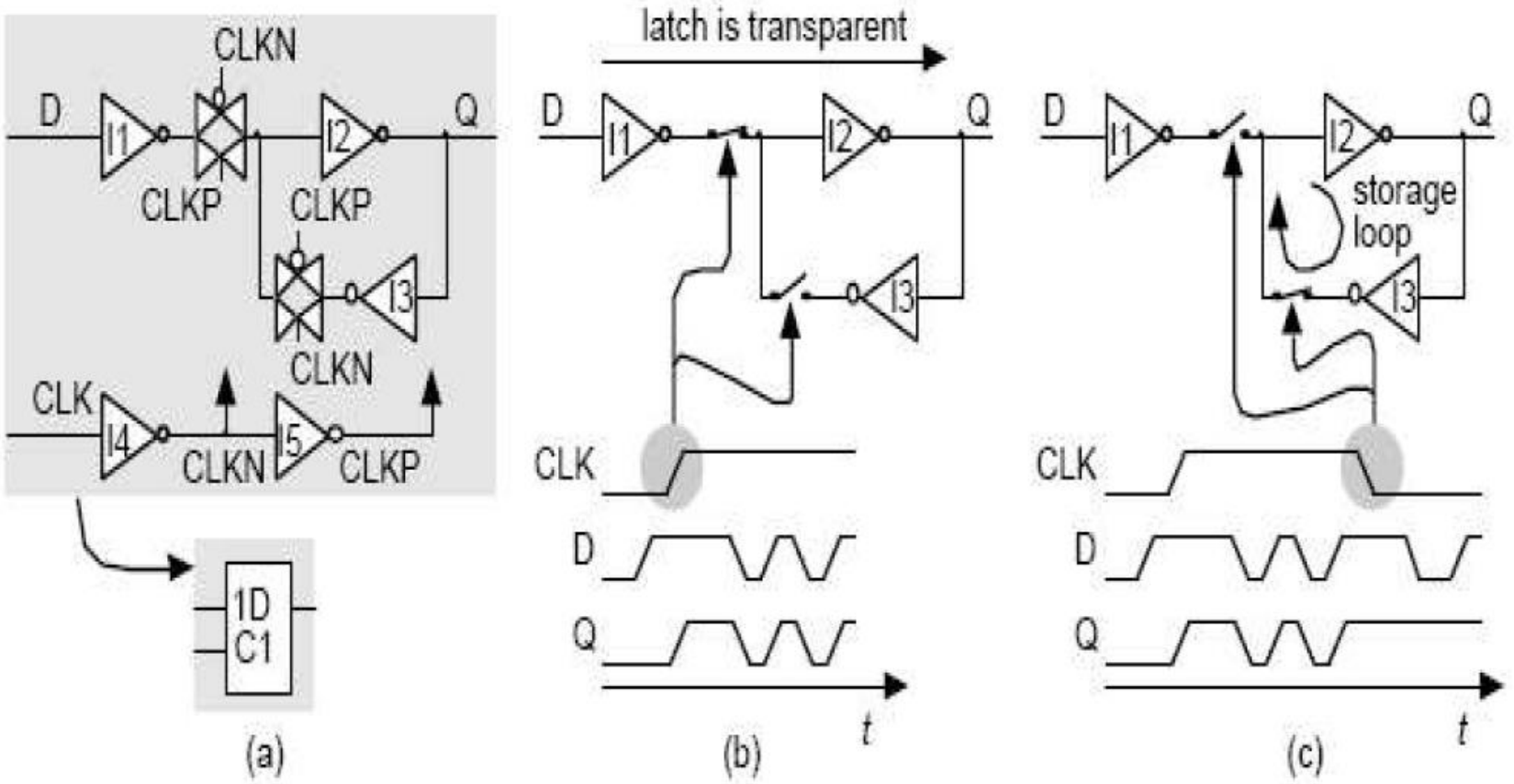


Transmission Gates

CMOS transmission gate (TG, TX gate, pass gate, coupler)

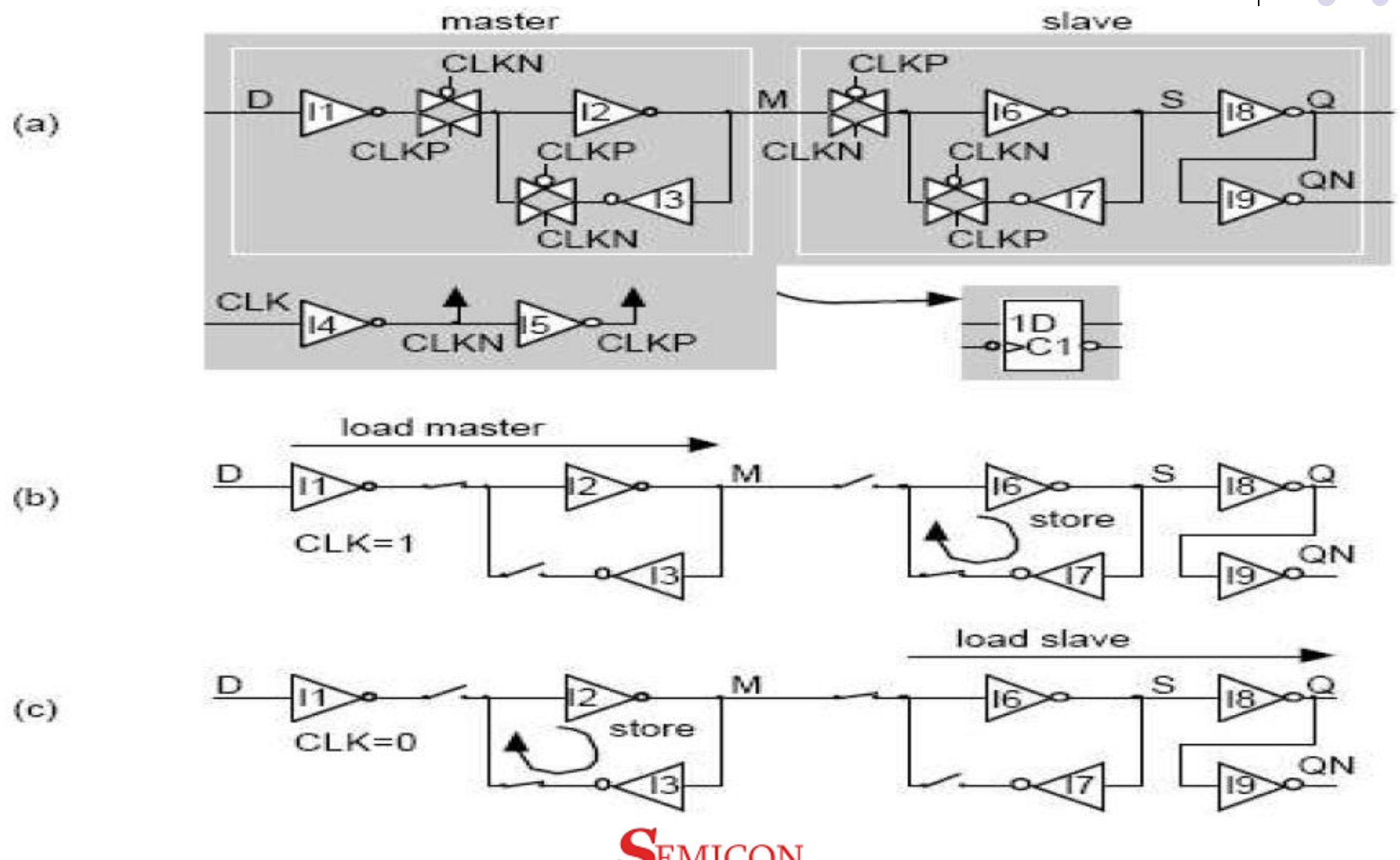


Sequential Logic Cells

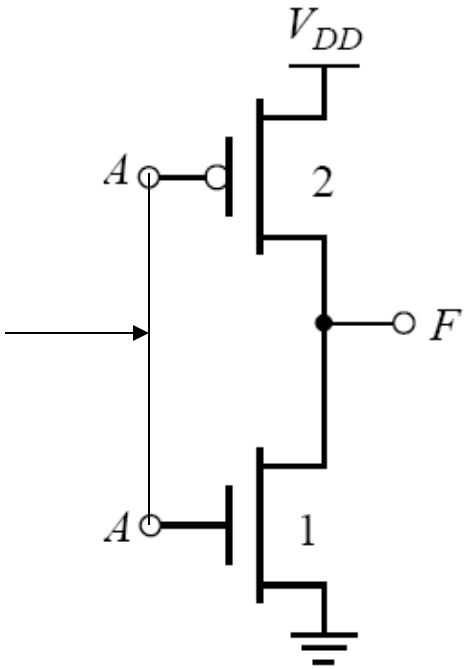


CMOS latch • **enable** • **transparent** • **static** • **sequential logic cell** • **storage** • **initial value**

CMOS flip-flop

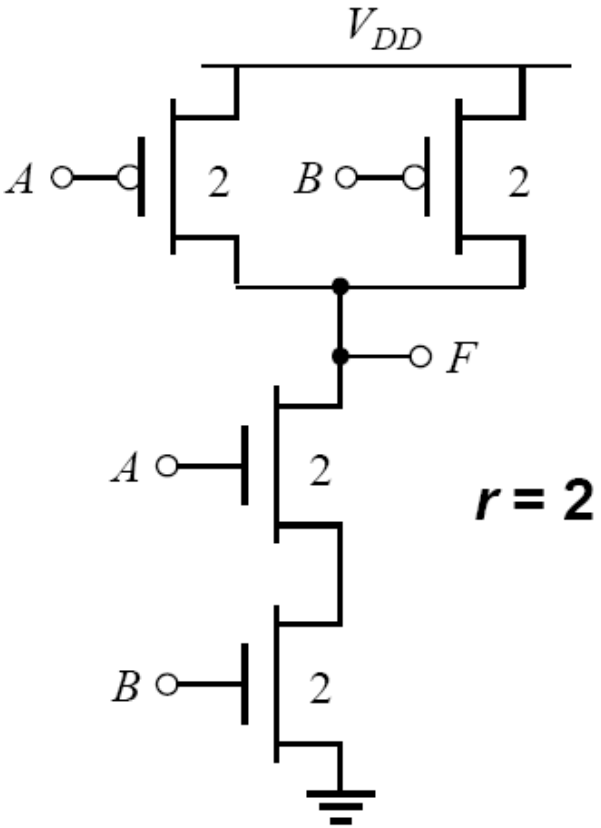


Example of Basic Gates



Inverter

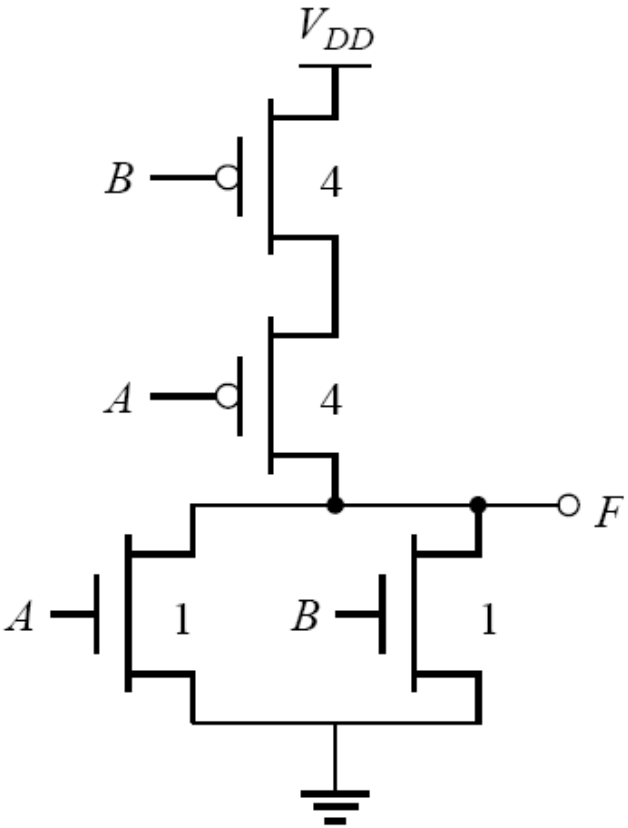
g = 1



2-input NAND

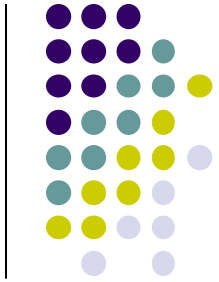
r = 2

g = 4/3



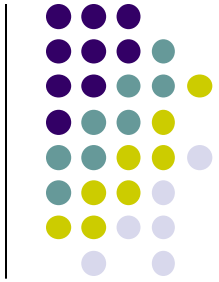
2-input NOR

g = 5/3



Specification Design

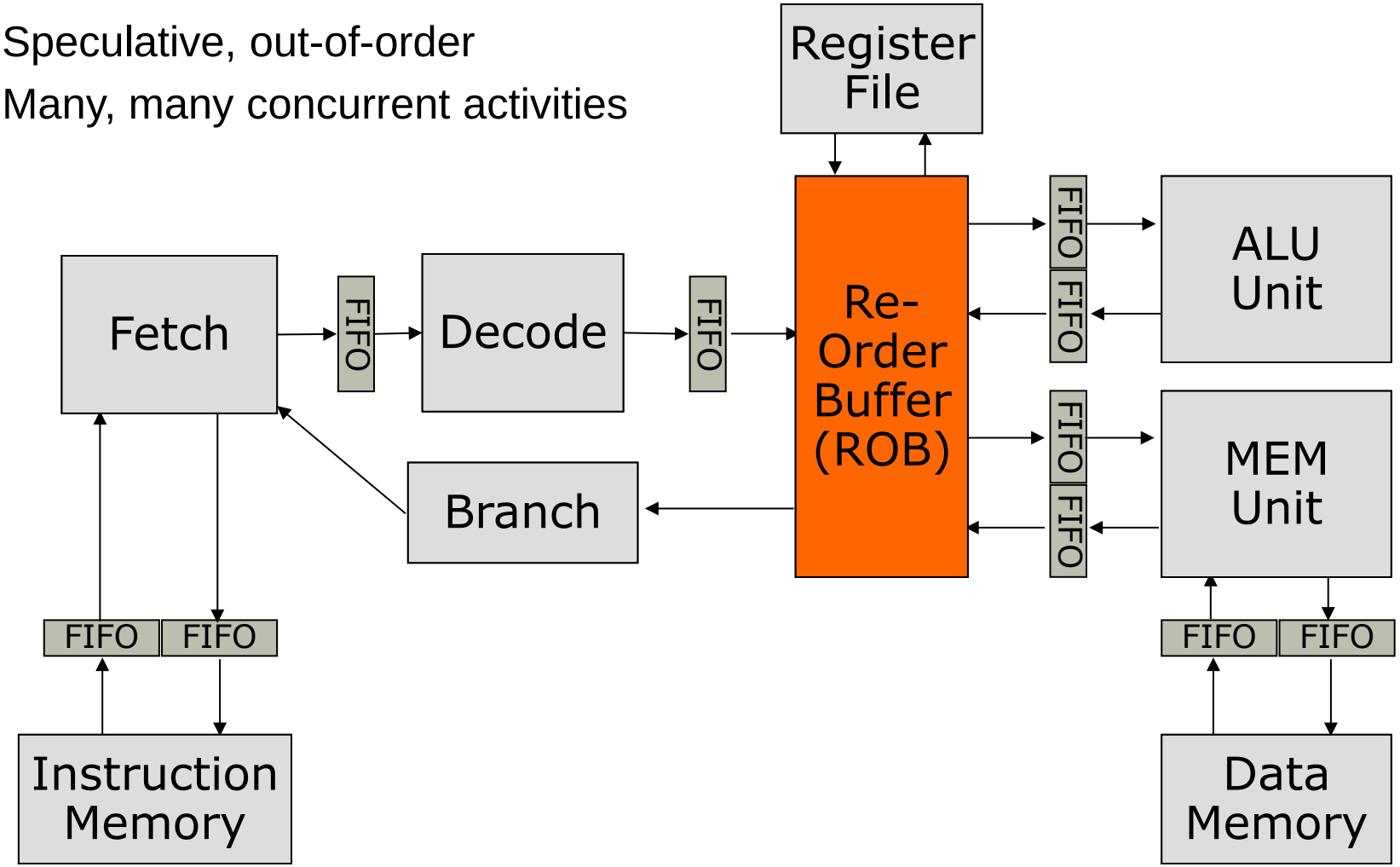
Finish Design Documents

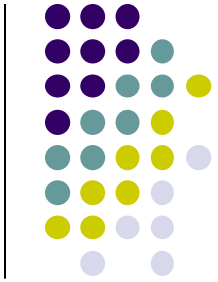


- Interface Specification
- Function Specification
- Design Details

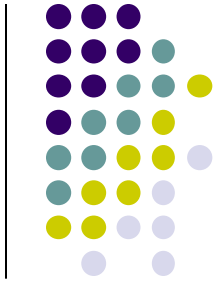
Detail block is finished

- Speculative, out-of-order
- Many, many concurrent activities



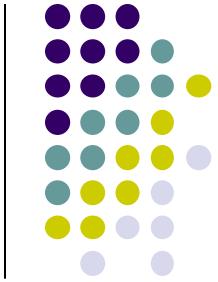


RTL Coding



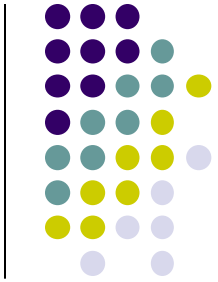
Simulation

Agenda



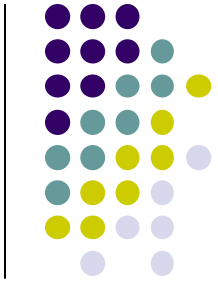
- Introduction
- How Logic Simulation Works
- Types of Simulation
  - Behavioral Simulation
  - Functional Simulation
    - Formal Verification
  - Static Timing Analysis
  - Gate-Level Simulation
  - Switch-Level Simulation
  - Transistor-Level Simulation

Introduction



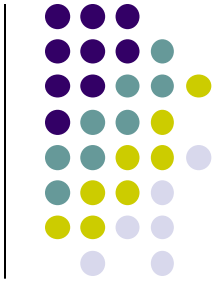
- Simulation is used at many stages during ASIC design.
- Initial prelayout simulations include logic cell delays but no interconnect delays.
- Estimates of capacitance may be included after completing logic synthesis, but only after physical design it is possible to perform an accurate postlayout simulation.

How Logic Simulation Works



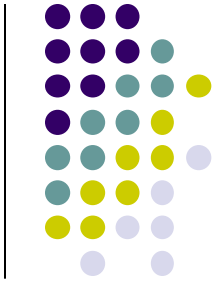
- The most common type of digital simulator is an eventdriven simulator.
- When a circuit node changes in value the time, the node, and the new value are collectively known as an event.
- The event is scheduled by putting it in an event queue or event list.
- When the specified time is reached, the logic value of the node is changed.
- The change affects logic cells that have this node as an input.
- All of the affected logic cells must be evaluated, which may add more events to the event list.

How Logic Simulation Works



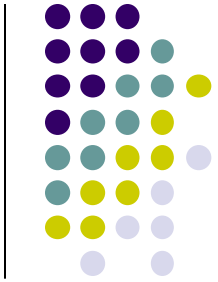
- The simulator keeps track of the current time, the current time step, and the event list that holds future events.
- For each circuit node the simulator keeps a record of the logic state and the strength of the source or sources driving the node.
- When a node changes logic state, whether as an input or as an output of a logic cell, this causes an event.
- An interpreted-code simulator uses the HDL model as data, compiling an executable model as part of the simulator structure, and then executes the model.
- This type of simulator usually has a short compile time but a longer execution time compared to other types of simulator.

Types of Simulation



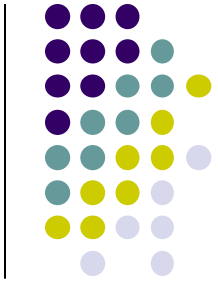
- Simulators are usually divided into the following categories or simulation modes.
  - Behavioral simulation
  - Functional simulation
  - Static timing analysis
  - Gate-level simulation
  - Switch-level simulation
  - Transistor-level or circuit-level simulation

Types of Simulation



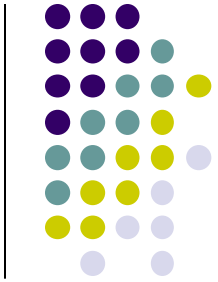
- This list is ordered from high-level to low-level simulation (high-level being more abstract, and low-level being more detailed).
- Proceeding from high-level to low-level simulation, the simulations become more accurate, but they also become progressively more complex and take longer to run.
- While it is just possible to perform a behavioral-level simulation of a complete system, it is impossible to perform a circuit-level simulation of more than a few hundred transistors.
- There are several ways to create an imaginary simulation model of a system.

Behavioral Simulation



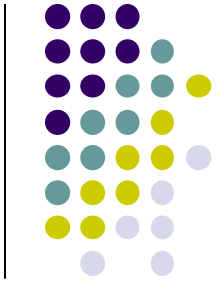
- The method which models large pieces of a system as black boxes with inputs and outputs.
- This type of simulation (often using VHDL or Verilog) is called **behavioral simulation**.
- Thus, behavioral simulation can only tell us if our design does work; it cannot tell us that real hardware will work.
- We use logic synthesis to produce a structural model from a behavioral model.

Functional Simulation



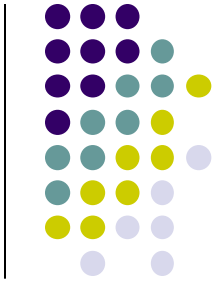
- To find the critical path using logic simulation requires simulating all possible input transitions and then shifting through the output to find the critical path.
- Vector-based simulation (or dynamic simulation) can show us that our design functions correctly hence the name **functional simulation**.
- Functional simulation ignores timing and includes unit-delay simulation, which sets delays to a fixed value (for example, 1 ns).
- However, functional simulation does not work well if we wish to find the critical path.Functional Simulation

Functional Simulation



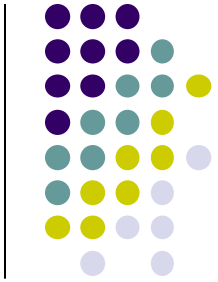
- Once a behavioral or functional simulation predicts that a system works correctly, the next step is to check the timing performance.
- At this point a system is partitioned into smaller ASICs and a timing simulation is performed for each ASIC separately (otherwise the simulation run times become too long).
- One class of timing simulators employs timing analysis that analyzes logic in a static manner, computing the delay times for each path, this is called **static timing analysis** because it does not require the creation of a set of test (or stimulus) vectors (difficult for a large ASIC).
- Timing analysis works best with synchronous systems whose maximum operating frequency is determined by the longest path delay between successive flip-flops.

Functional Simulation



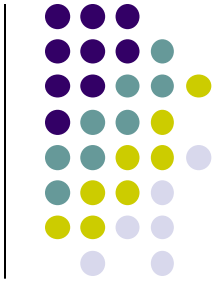
- The path with the longest delay is the critical path.
- Logic simulation or gate-level simulation can also be used to check the timing performance of an ASIC.
- In a gate-level simulator a logic gate or logic cell (NAND, NOR, and so on) is treated as a black box modeled by a function whose variables are the input signals.
- The function may also model the delay through the logic cell.
- Setting all the delays to unit value is the equivalent of functional simulation.
- Timing information for most gate-level simulators is calculated once, before simulation, using a delay calculator.
- This works as long as the logic cell delays and signal ramps do not change.

Functional Simulation



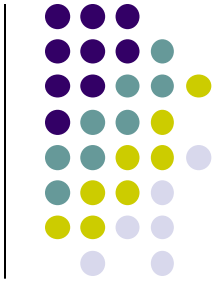
- In addition to pin-to-pin timing differences there is a timing difference depending on state.
- If the timing simulation provided by a black-box model of a logic gate is not accurate enough?
- The next, more detailed, level of simulation is **switchlevel simulation** which models transistors as switches on or off.
- Switch-level simulation can provide more accurate timing predictions than gate-level simulation, but without the ability to use logic-cell delays as parameters of the models.
- The most accurate, but also the most complex and timeconsuming, form of simulation is **transistor-level simulation**.
- A transistor-level simulator requires models of transistors, describing their nonlinear voltage and current characteristics.

Formal Verification



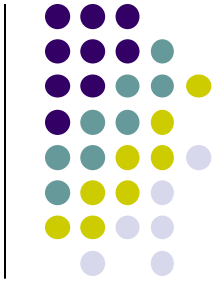
- Using logic synthesis we move from a behavioral model to a structural model.
- How are we to know (other than by trusting the logic synthesizer) that the two representations are the same?
- We have already seen that we may have to alter the original reference model because the HDL acceptable to a synthesis tool is a subset of HDL acceptable to simulators.
- Formal verification can prove, in the mathematical sense, that two representations are equivalent.
- If they are not, the software can tell us why and how two representations differ.

Static Timing Analysis

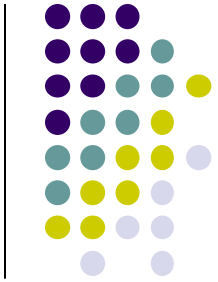


- A timing analyzer answers the question: What is the longest delay in my circuit?
- The timing analyzer gives us only the critical path and its delay.
- A timing analyzer does not give us the input vectors that will activate the critical path.
- In fact input vectors may not exist to activate the critical path.
- The Paths which are impossible to activate, are known as false paths.
- Timing analysis is essential to ASIC design but has limitations.
- A timing-analysis tool is more logic calculator than logic simulator.

Gate-Level Simulation

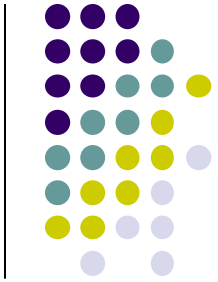


- The simulator is adding capacitance to the outputs of each of the logic cells to model the parasitic net capacitance (interconnect capacitance or wire capacitance) that will be present in the physical layout.
- The model that predicts these values is known as a wireload model, wire-delay model, or interconnect model.
- Changing the wire-load model parameters to zero and repeating the simulation changes the critical-path delay, which agrees exactly with the logic-synthesizer timing analysis.
- This emphasizes that the net capacitance may contribute a significant delay.



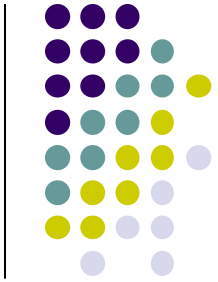
Logic Synthesis

Why Synthesis?



- Increasing complexity of designs.
  - Multi-million gate designs
- Time to market.
- Multiple target technologies.
- Conflicting constrains.
- Correct by construction.

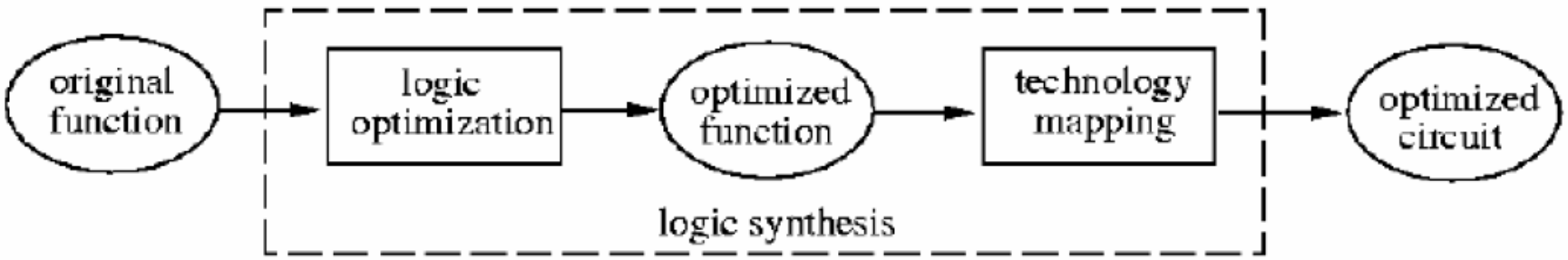
Logic Synthesis



- Logic synthesis provides a link between an HDL and a netlist similarly to the way that a C compiler provides a link between C code and machine language.
- C was developed for use with compilers, but HDLs were not developed for use with logic-synthesis tools.
- Verilog was designed as a simulation language.
- VHDL was designed as a documentation and description language.
- Both Verilog and VHDL were developed in the early 1980s, well before the introduction of commercial logic synthesis software.

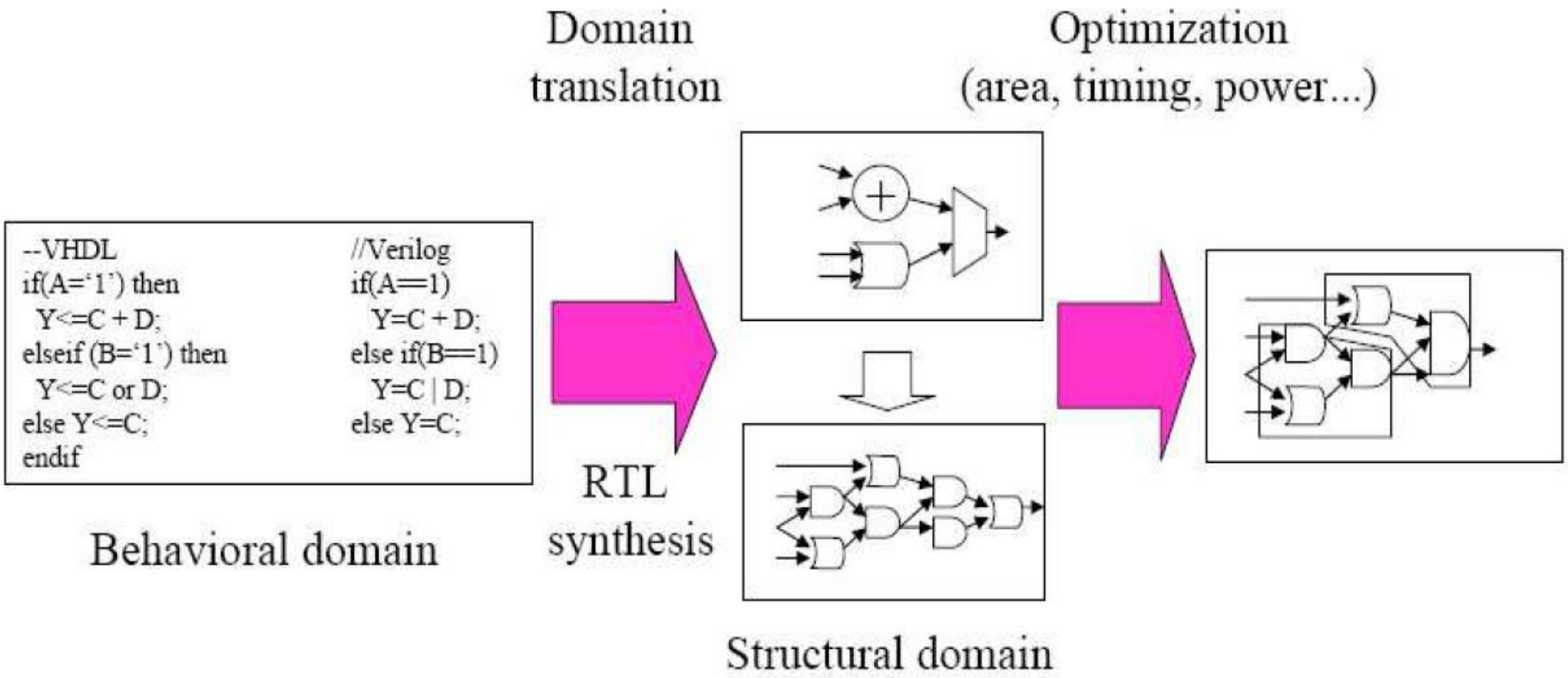
Logic Synthesis

- Logic Synthesis optimize the implementation cost of Boolean logic.
- Cost:
  - area, delay, power.
- Optimization can be:
  - Technology independent optimization (logic optimization)
  - Technology dependent optimization (technology mapping)



HDL Synthesis

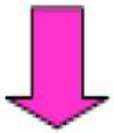
Synthesis = Domain Translation + Optimization



For example

$$f1 = abcd + abce + \bar{a}bcd + a\bar{b}\bar{c}d + \bar{a}c + cdf + ab\bar{c}\bar{d}e + a\bar{b}cdf$$

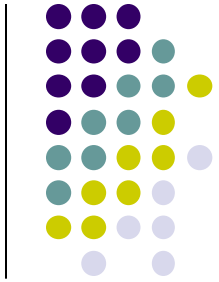
$$f2 = bdg + \bar{b}dfg + \bar{b}\bar{d}g + b\bar{d}eg$$



$$f1 = c\,(\bar{a} + x) + a\bar{c}\bar{x}$$

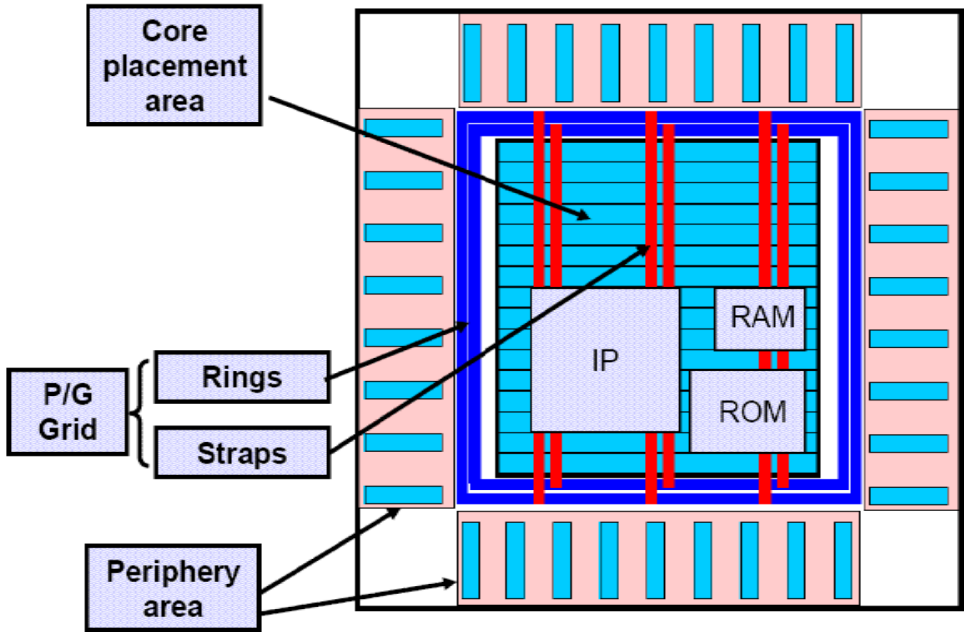
$$f2 = gx$$

$$x = d\,(b + f) + \bar{d}\,(\bar{b} + e)$$



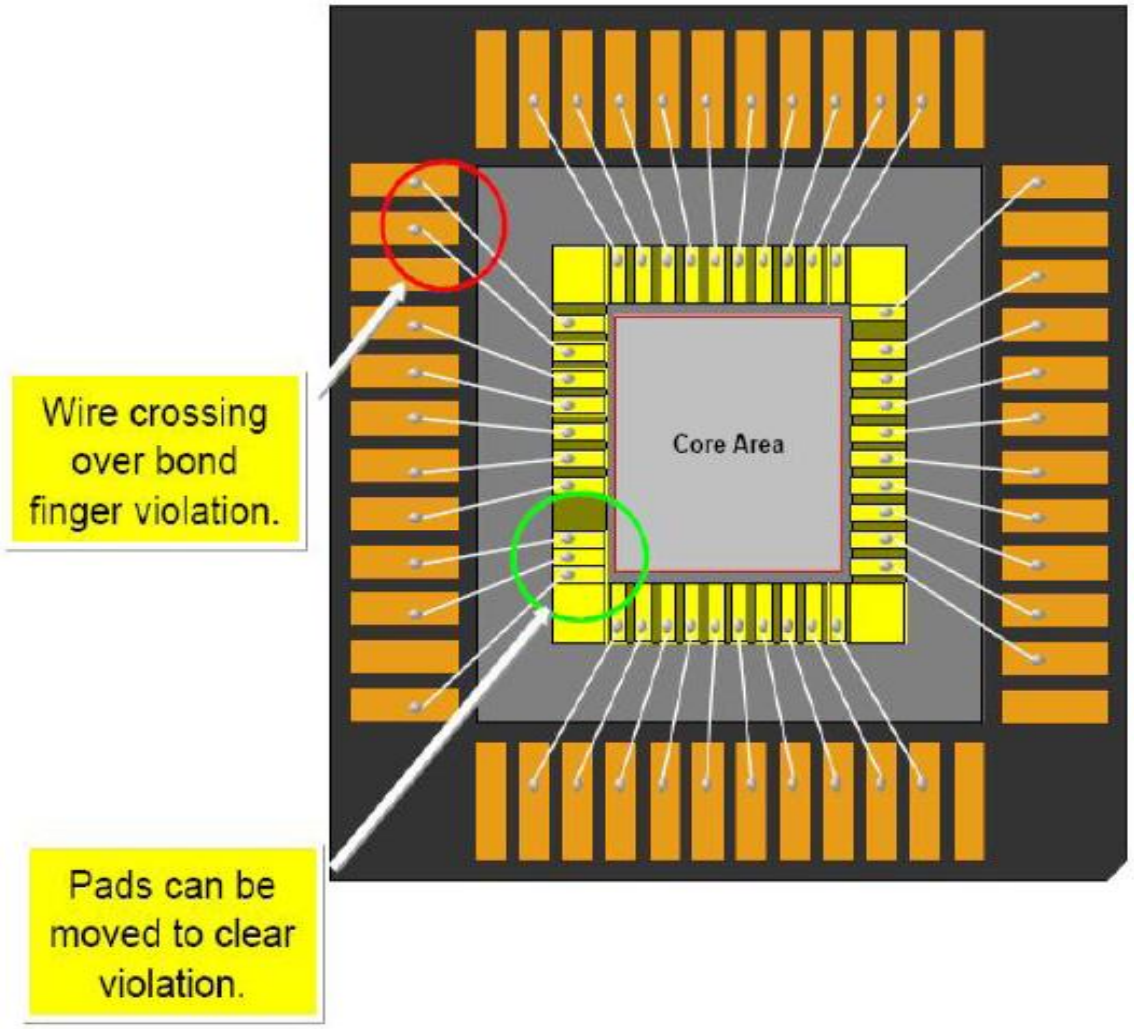
Floorplanning

Design Must Be Floorplanned Before P&R



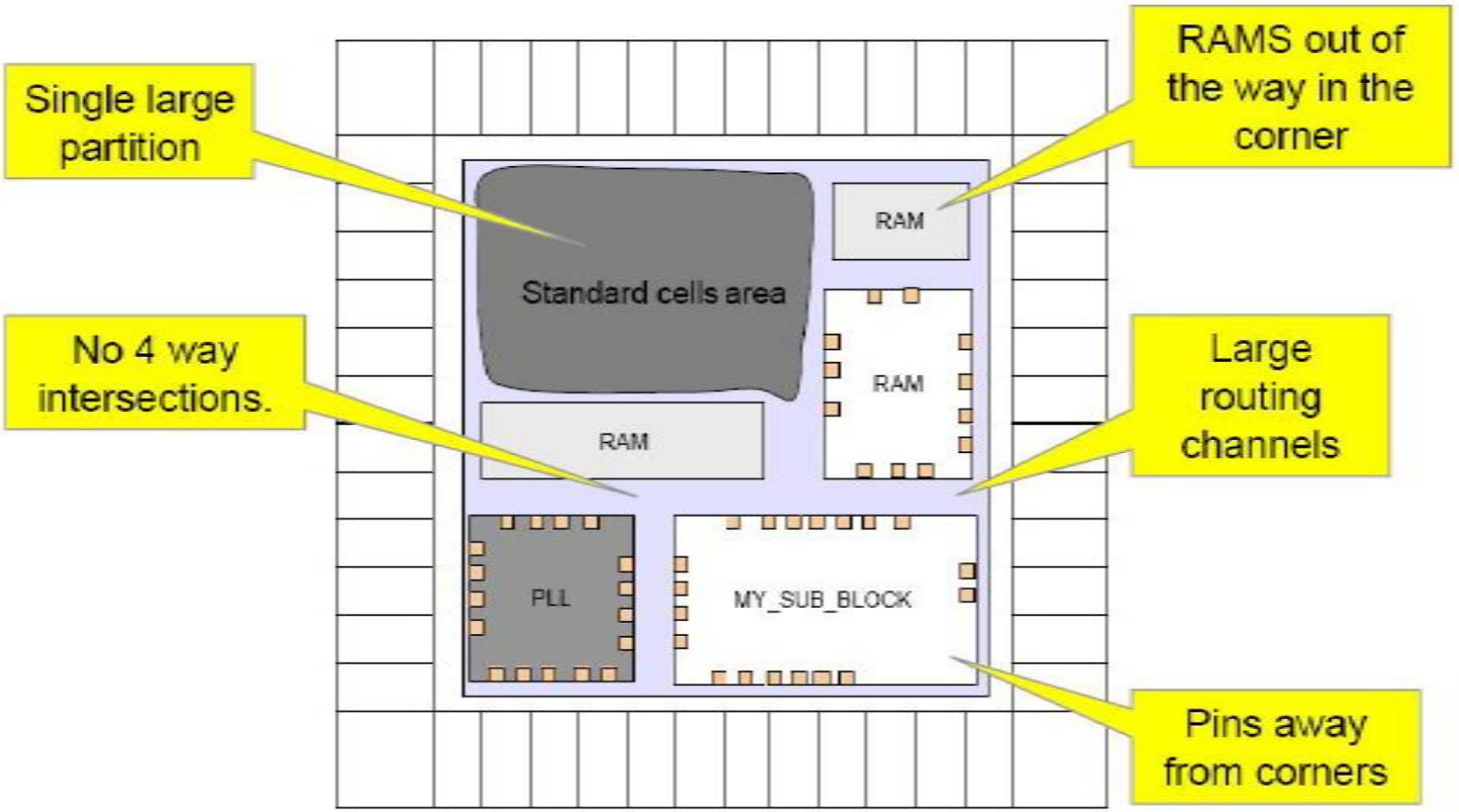
- Floorplan of design:
  - Core area defined with large macros placed
  - Periphery area defined with I/O macros placed
  - Power and Ground Grid (Rings and Straps) established
- Utilization:
  - The percentage of the core that is used by placed standard cellsand macros
  - Goal of 100%, typically 80-85%

I/O Placement and Chip Package Requirements



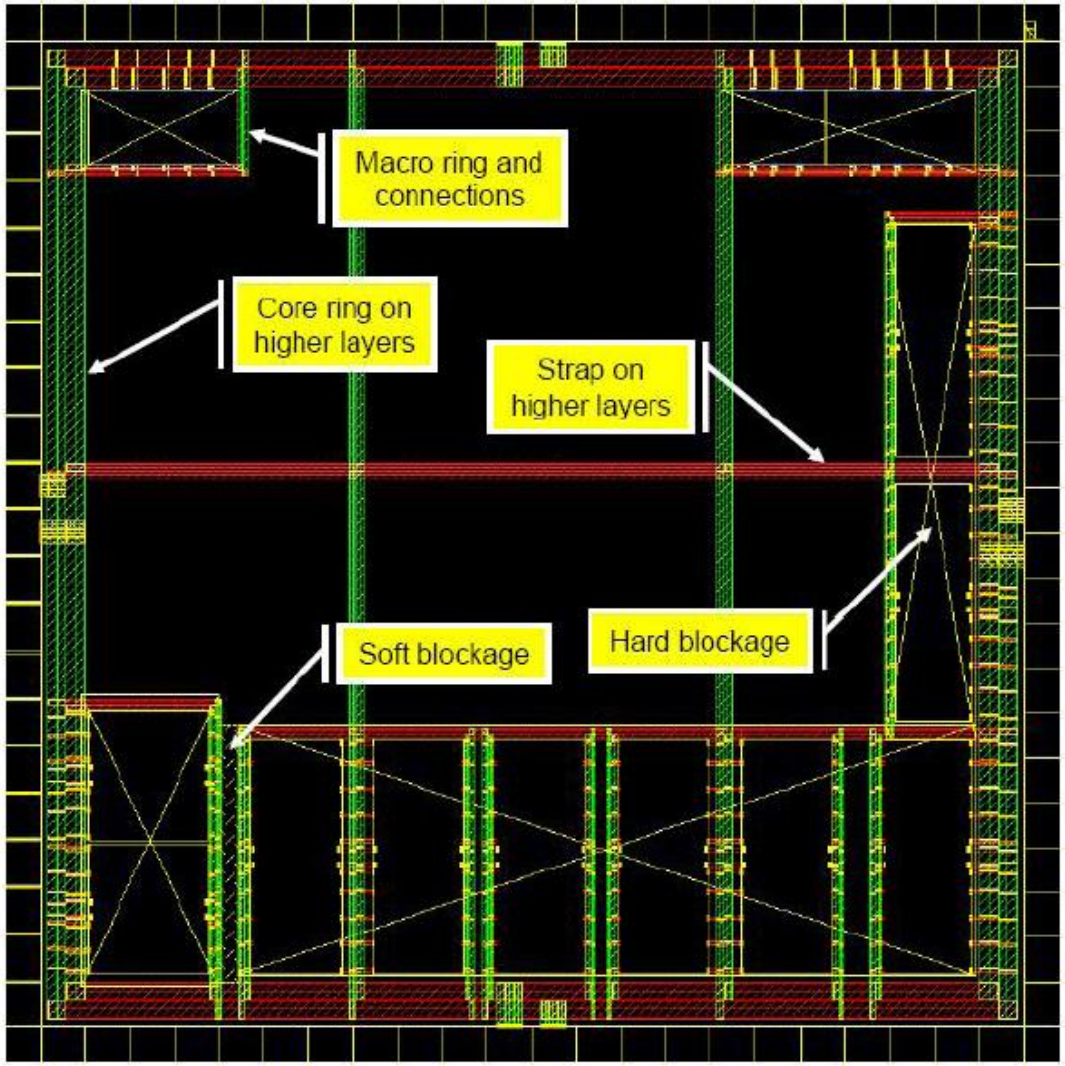
- Some Bond Wire requirements:
  - No Crossing
  - Minimum Spacing
  - Maximum Angle
  - Maximum Length

Guidelines for a Good Floorplan



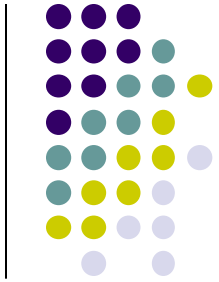
A few quick iterations of place and route with timing checks may reveal the need for a different floorplan

Defining the Power/Ground Grid and Blockages



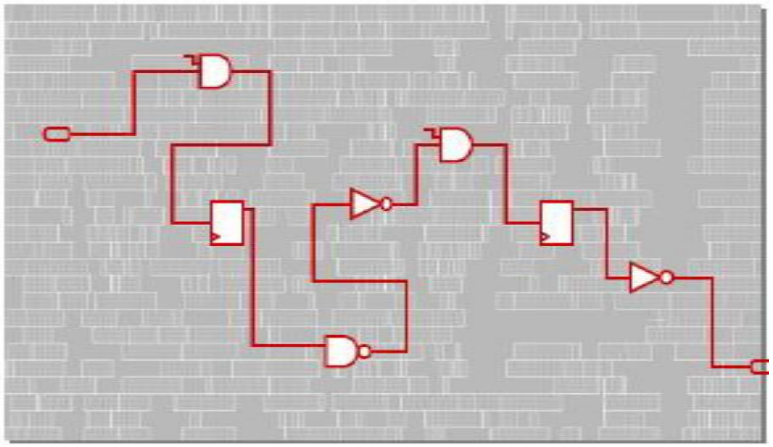
- Purpose of Grid is to take the VDD and VSS received from the I/O area and distribute it over the core area

- Blockages can also be added in the floorplan to prohibit standards cells from being placed in those areas

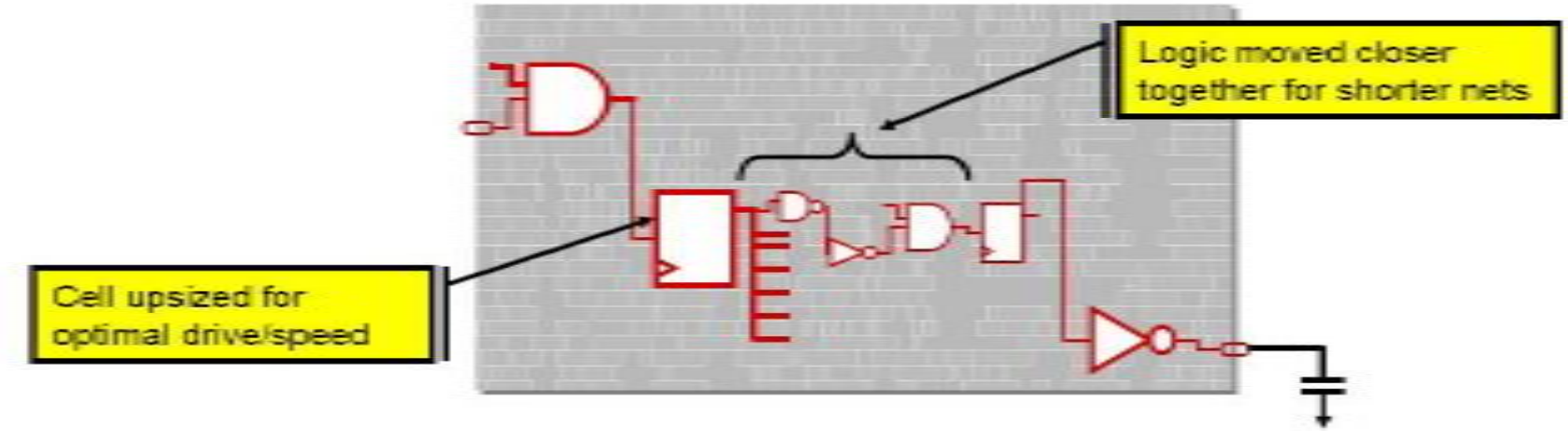


Placement

Timing Driven Placement

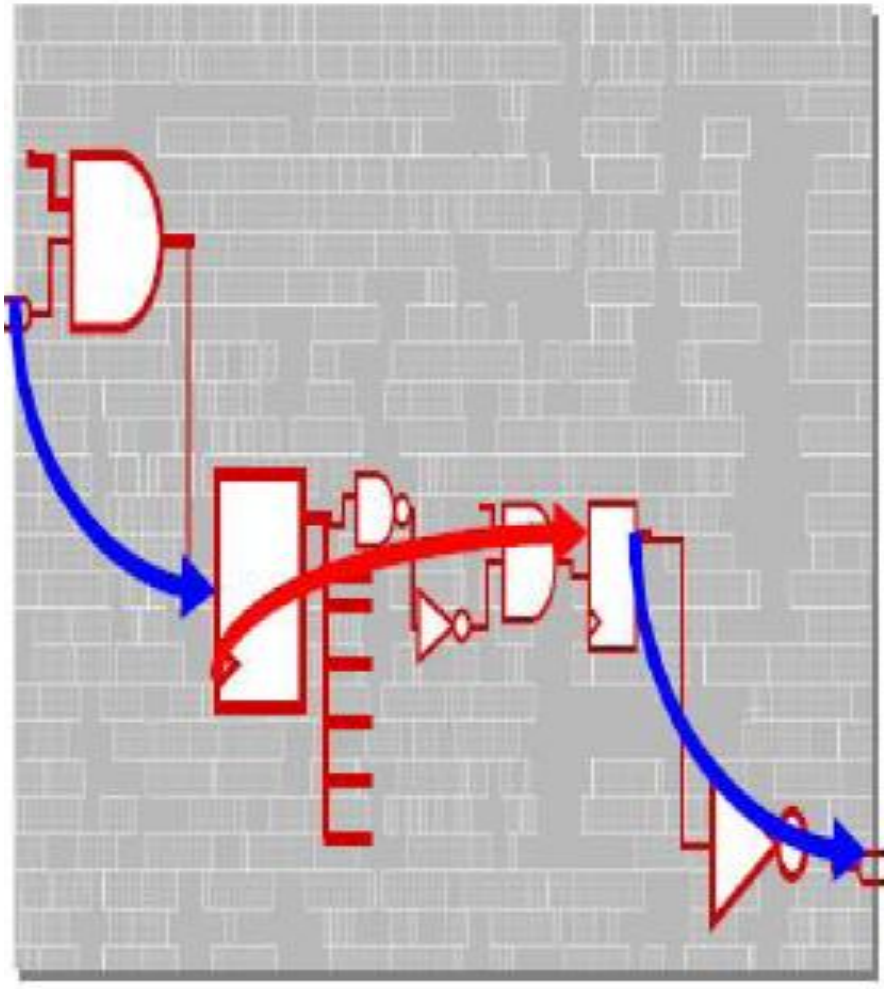


- Tools optimizes, places, and routes the logic gates to meet all timing constraints
- Balancing design requirements
  - Timing
  - Area
  - Power
  - Signal Integrity



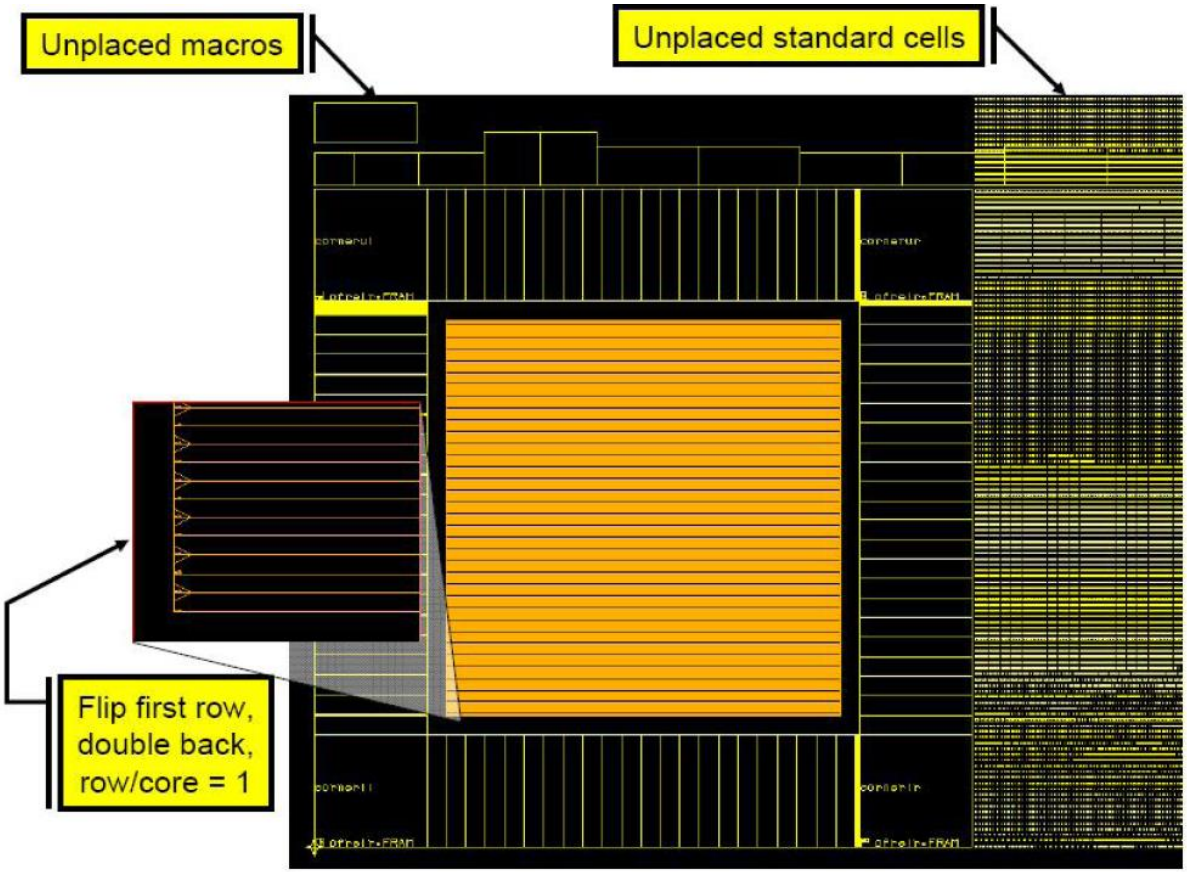
Timing Constraints

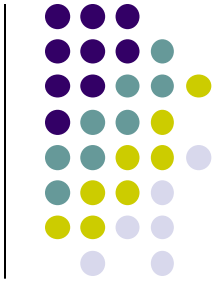
- Tool needs constraints to understand the timing intentions
  - Arrival time of inputs
  - Required arrival time at outputs
  - Clock period



Timing Driven Placement

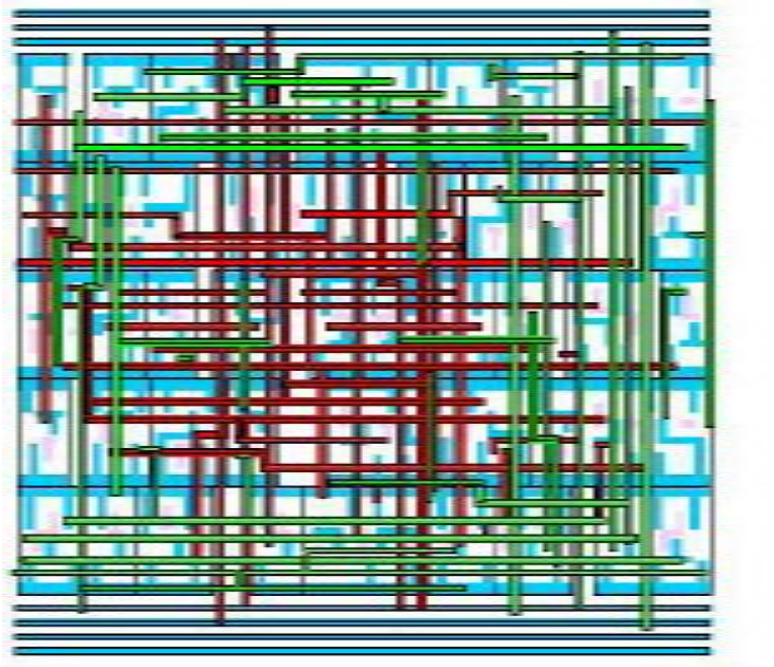
- Timing Driven Placement places critical path cells close together to reduce net RC
- Prior to routing, RC are based on Virtual Routes
- What if critical paths do not meet timing constraints with placement?





Routing

Routing

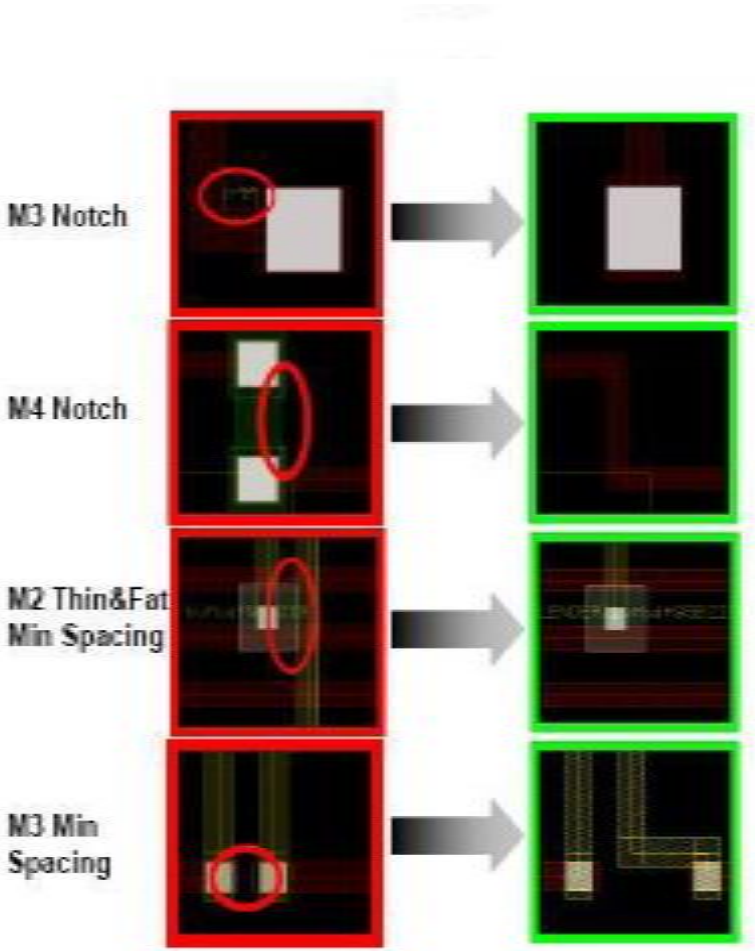


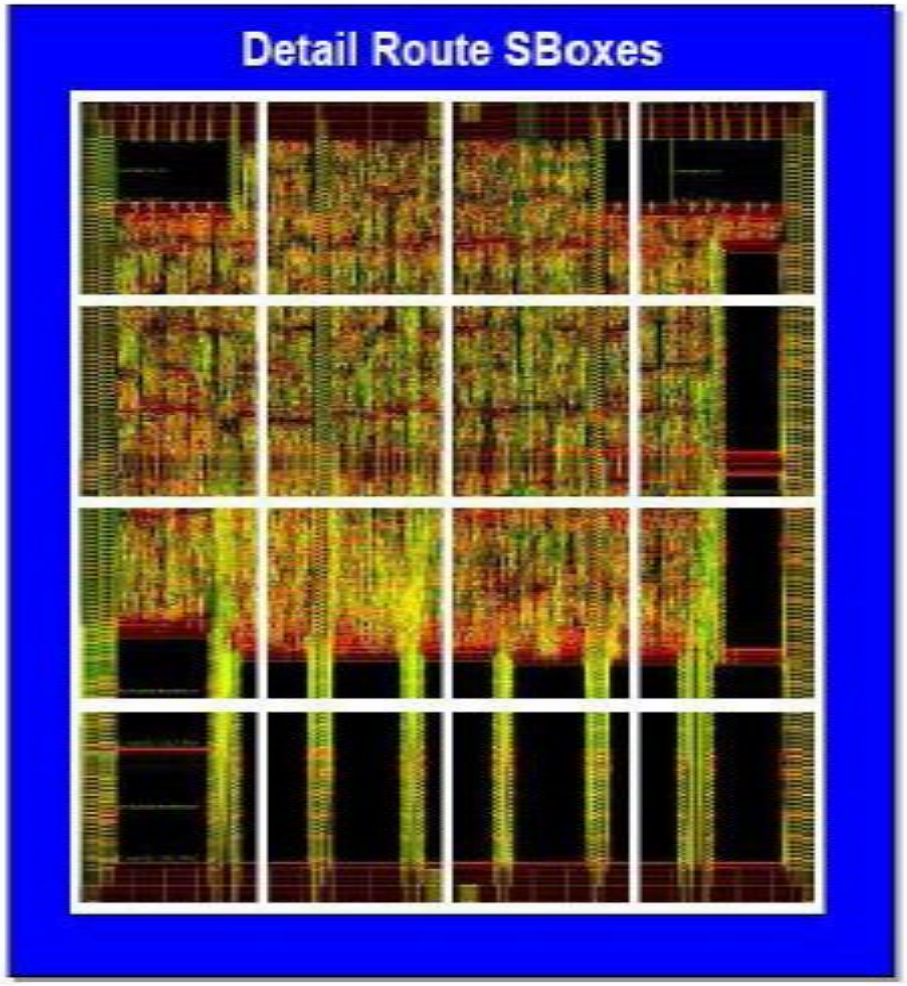
Critical Net

Non-Critical Net

- Routing along the timing
  - critical path is given priority
  - Creates shorter, faster connections
- Non-critical paths are routed around critical areas
  - Reduces routing congestion problems for critical paths
  - Does not adversely impact timing of non-critical paths

Detailed Routing





Conclusion

- Designing a sophisticated VLSI design is complicated
  - Number of transistors combined with challenges at each processing node
- Design time can be over 1 year from system conception to fabricated chips
- No one person or design team could manually design and complete a system in a time frame to be market competitive

Task of Designing, Placing, Routing, and Fabricating Complex VLSI Designs Must Become Automated

Câu Hỏi & Trả Lời

icdesign1@semiconvn.com

Password: yovi2008

SEMICON
THINKING OUT OF THE BOX