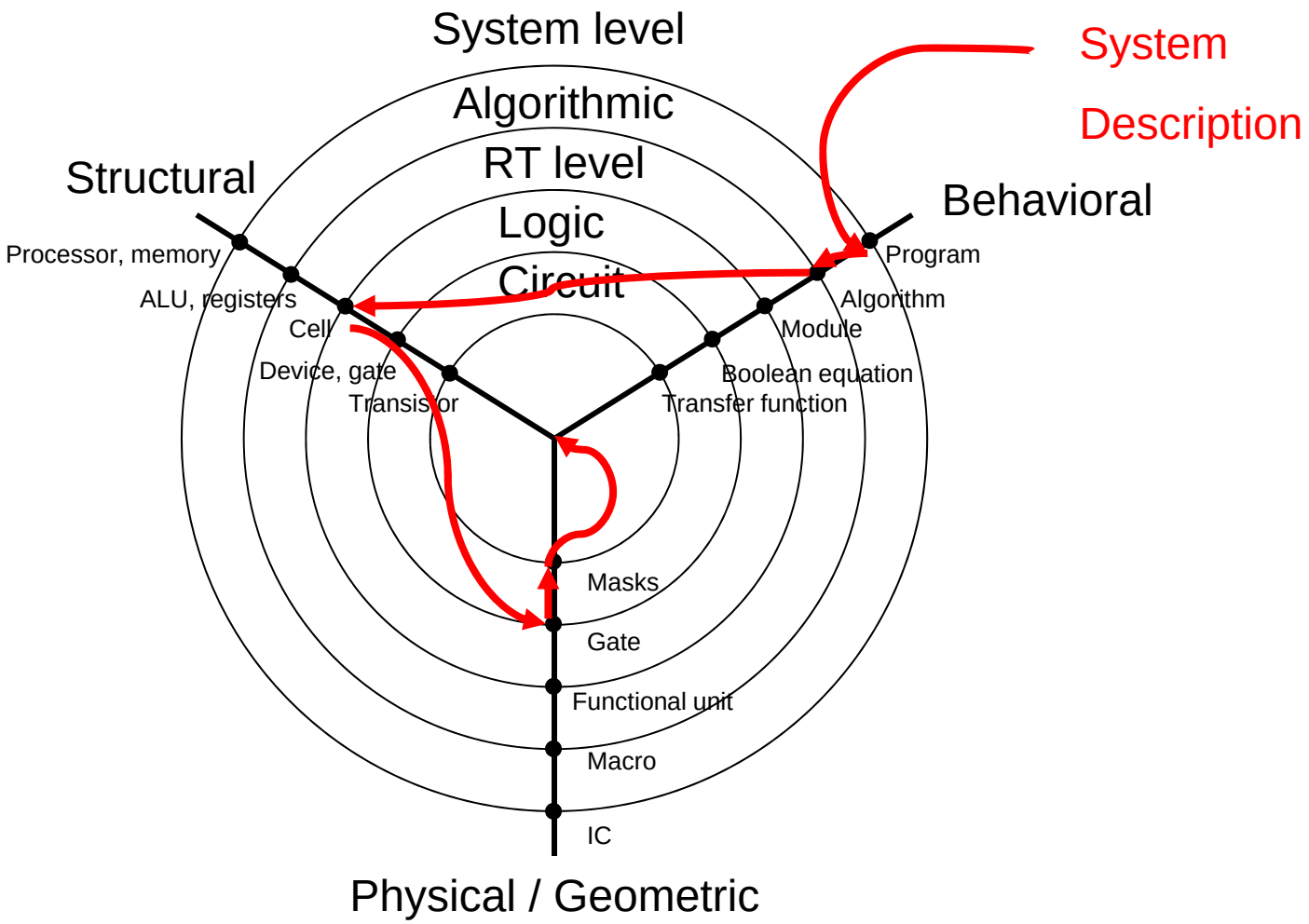
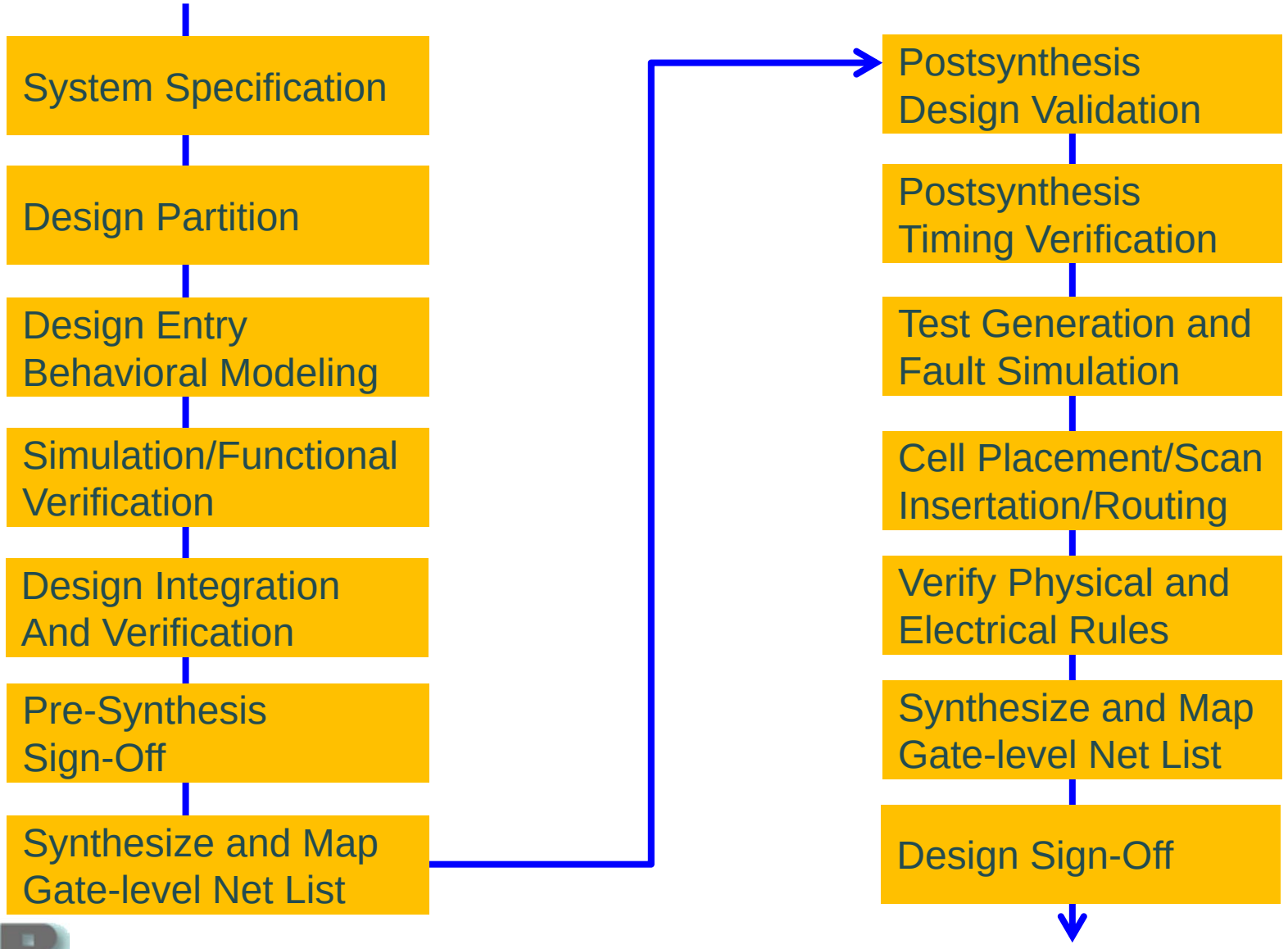
Design Methodology: A walkthrough

Ramon Canal
2013

Design domains (Gajski i Kuhn)



Design Methodology: Big Picture



Design Specification

- Written statement of functionality, timing, area, power, testability, fault coverage, etc.
- Functional specification methods:
  - State Transition Graphs
  - Timing Charts
  - Algorithm State Machines (like flowcharts)
  - HDLs (Verilog and VHDL)

FIB

Design Partition

- Partition to form an Architecture
  - Interacting functional units
    - Control vs. datapath separation
    - Interconnection structures within datapath
    - Structural design descriptions
  - Components described by their behaviorals
    - Register-transfer descriptions
  - Top-down design method exploiting hierarchy and reuse of design effort

FIB

Design Entry

- Primary modern method: hardware description language
  - Higher productivity than schematic entry
  - Inherently easy to document
  - Easier to debug and correct
  - Easy to change/extend and hence experiment with alternative architectures
- Synthesis tools map description into generic technology description
  - E.g., logic equations or gates that will subsequently be mapped into detailed target technology
  - Allows this stage to be technology independent (e.g., FPGA LUTs or ASIC standard cell libraries)
- Behavioral descriptions are how it is done in industry today

FIB

Simulation and Functional Verification

- Simulation vs. Formal Methods
- Test Plan Development
  - What functions are to be tested and how
  - Testbench Development
    - Testing of independent modules
    - Testing of composed modules
  - Test Execution and Model Verification
    - Errors in design
    - Errors in description syntax
    - Ensure that the design can be synthesized
  - The model must be VERIFIED before the design methodology can proceed

FIB

Design Integration and Verification

- Integrate and test the individual components that have been independently verified
- Appropriate testbench development and integration
- Extremely important step and one that is often the source of the biggest problems
  - Individual modules thoroughly tested
  - Integration not as carefully tested
  - Bugs lurking in the interface behavior among modules!

FIB

Presynthesis Sign-off

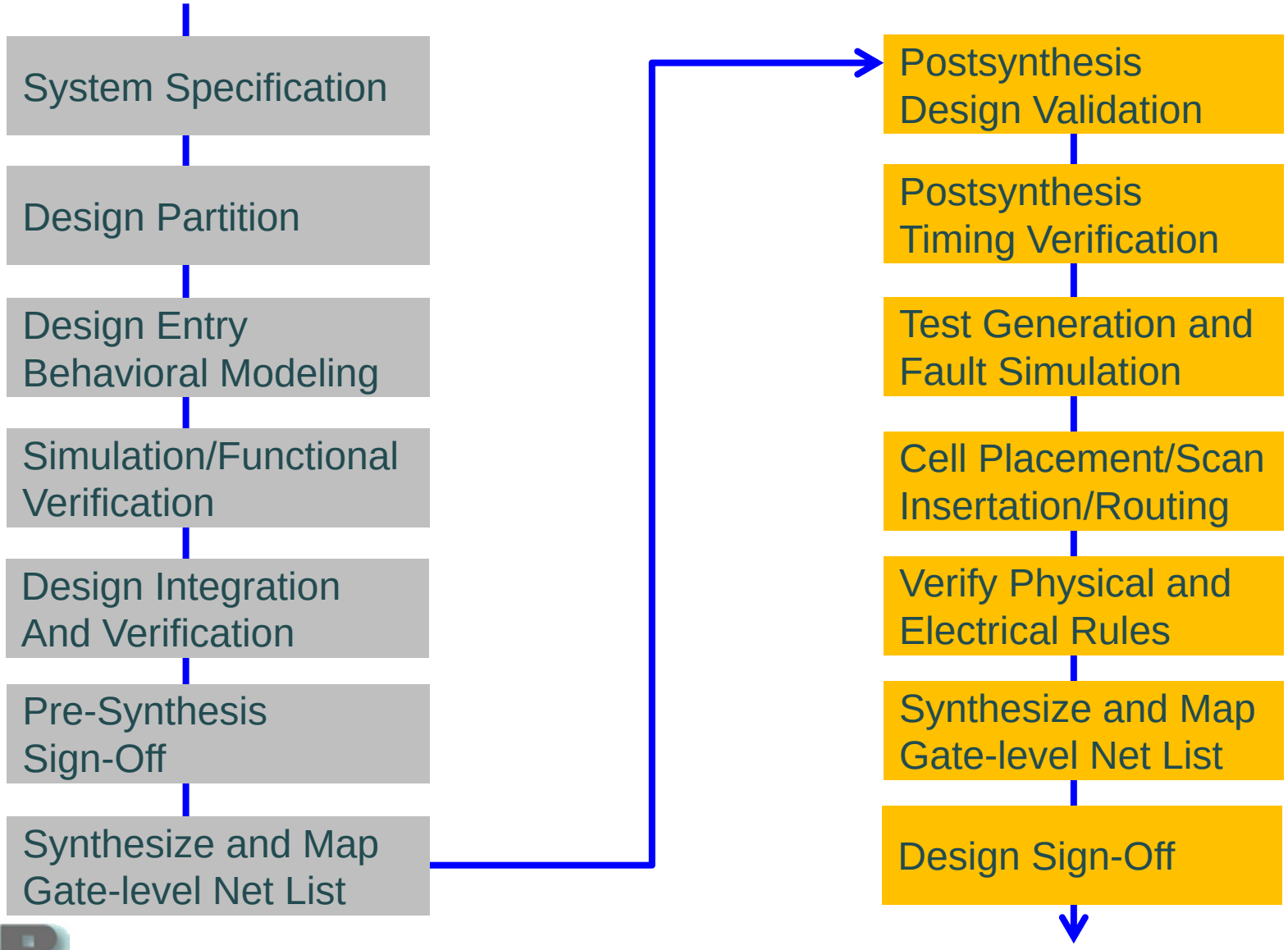
- Demonstrate full functionality of the design
- Make sure that the behavior specification meets the design specification
  - Does the demonstrated input/output behavior of the HDL description represent that which is expected from the original design specification
- Sign-off only when all functional errors have been eliminated

Gate-Level Synthesis and Technology Mapping

- Once all syntax and functional errors have been eliminated, synthesize the design from the behavior description
  - Optimized Boolean description
  - Map onto target technology
- Optimizations include
  - Minimize logic
  - Reduce area
  - Reduce power
  - Balance speed vs. other resources consumed
- Produces netlist according to target technology (standard cells, FPGA, …)

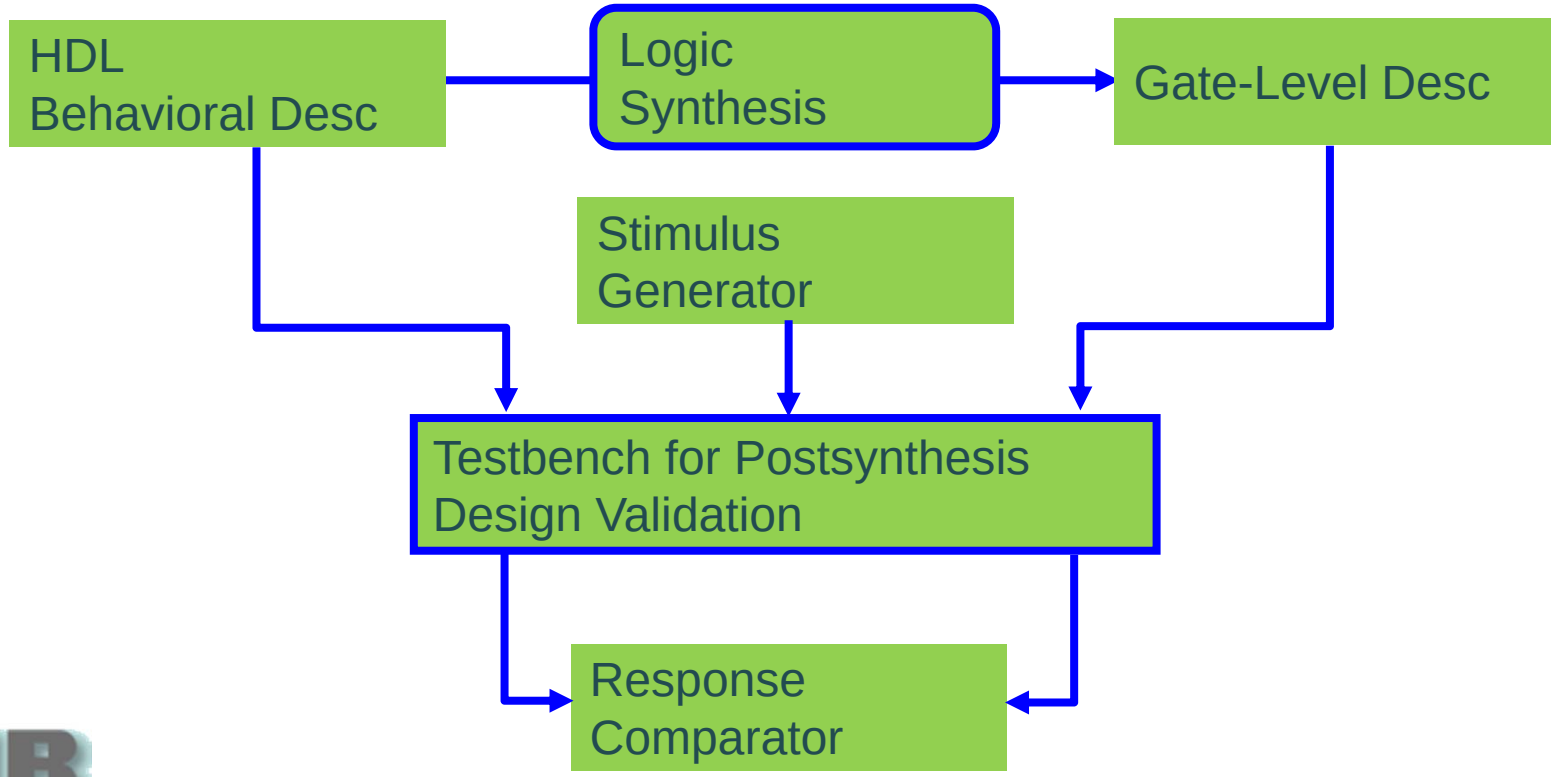
FIB

Design Methodology: Big Picture



Postsynthesis Design Validation

- Does gate-level synthesized logic implement the same input-output function as the HDL behavioral description?



Postsynthesis Timing Verification

- Are the timing specifications met?
- Are the speeds adequate on the critical paths?
  - Can’t accurately be determined until actual physical layout is understood and analyzed—length of wires, relative placement of sources and sinks, number of switch matrix crosspoints traversed, etc.
- Resynthesis may be required to achieve timing goals
  - Resize transistors
  - Modify architecture
  - Choose a different target device or technology

FIB

Test Generation and Fault Simulation

- This is NOT about debugging the design!
  - Design should be correct at this stage, so …
- Determine set of test vectors to test for inherent fabrication flaws
  - Need a quick method to sort out the bad from the good chips
  - More exhaustive testing may be necessary for chips that pass the first level
  - More relevant for ASIC design than FPGAs
    - Avoiding this step is one of the advantages of using the FPGA approach
- Fault simulation is used to determine how complete are the test vectors

Placement and Routing

- ASIC Standard Cells
  - Select the cells and placement them on the mask
  - Interconnect the placed cells
  - Choose implementation scheme for critical signals
    - E.g., Clock distribution trees to minimize skew
  - Insert scan paths
- FPGAs
  - Placing functions into particular CLBs/Slices and committing interconnections to particular wires in the switch matrix

FIB

Physical and Electrical Design Rule Check

- Applies to ASICs primarily
  - Are mask geometries correct to insure high probability of successful fabrication?
  - Fan-outs correct? Crosstalk signals within specification? Current drops within specification? Noise levels ok? Power dissipation acceptable?
- Many of these issues are not significant at a chip level for an FPGA but may be an issue for the system that incorporates the FPGA

Parasitic Extraction

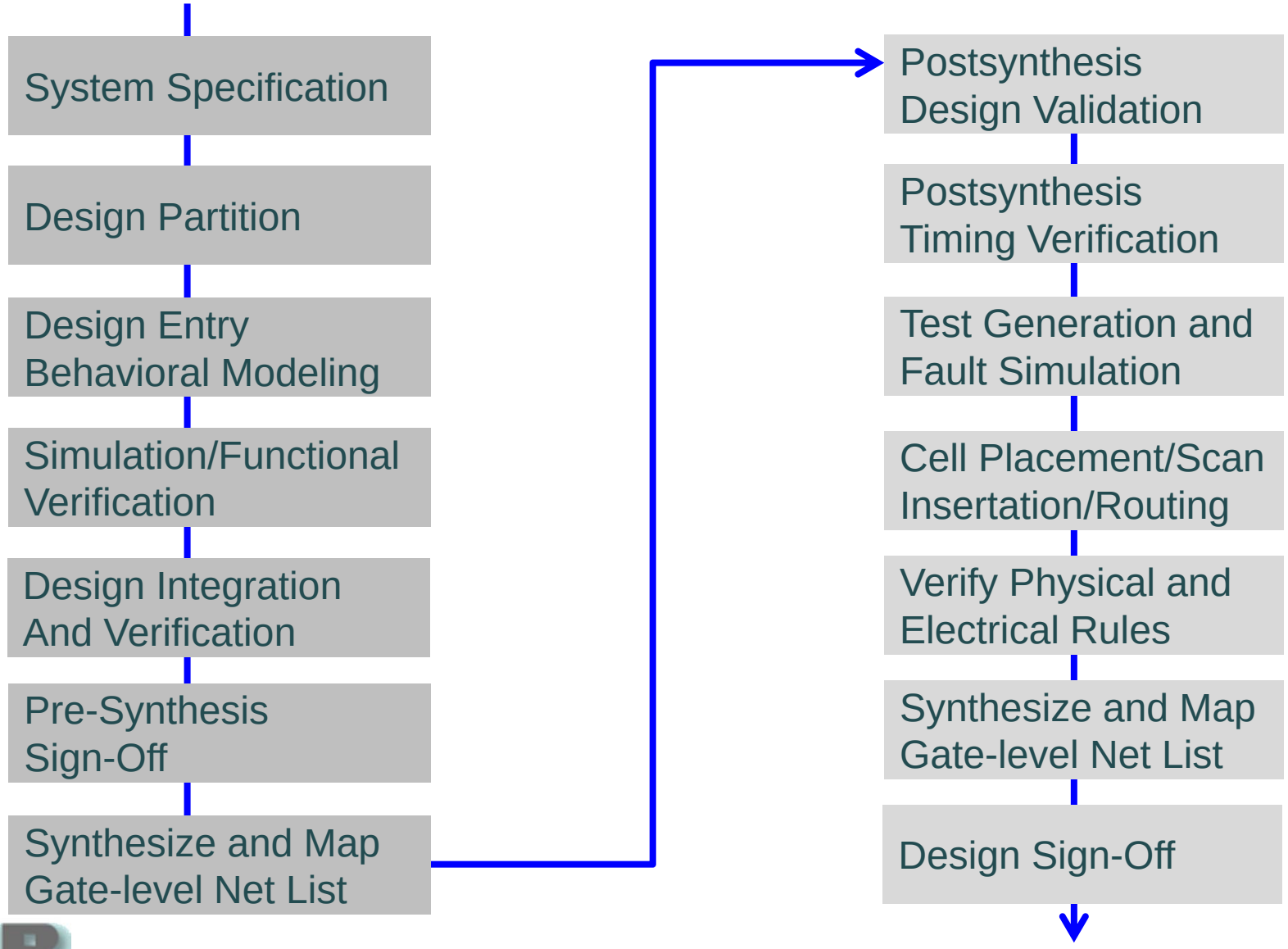
- Extract geometric information from design to determine capacitance
- Yields a much more realistic model of signal performance and delay
- Are the speed (timing) and power goals of the design still met?
- Could trigger another redesign/resythesize cycle if not met

FIB

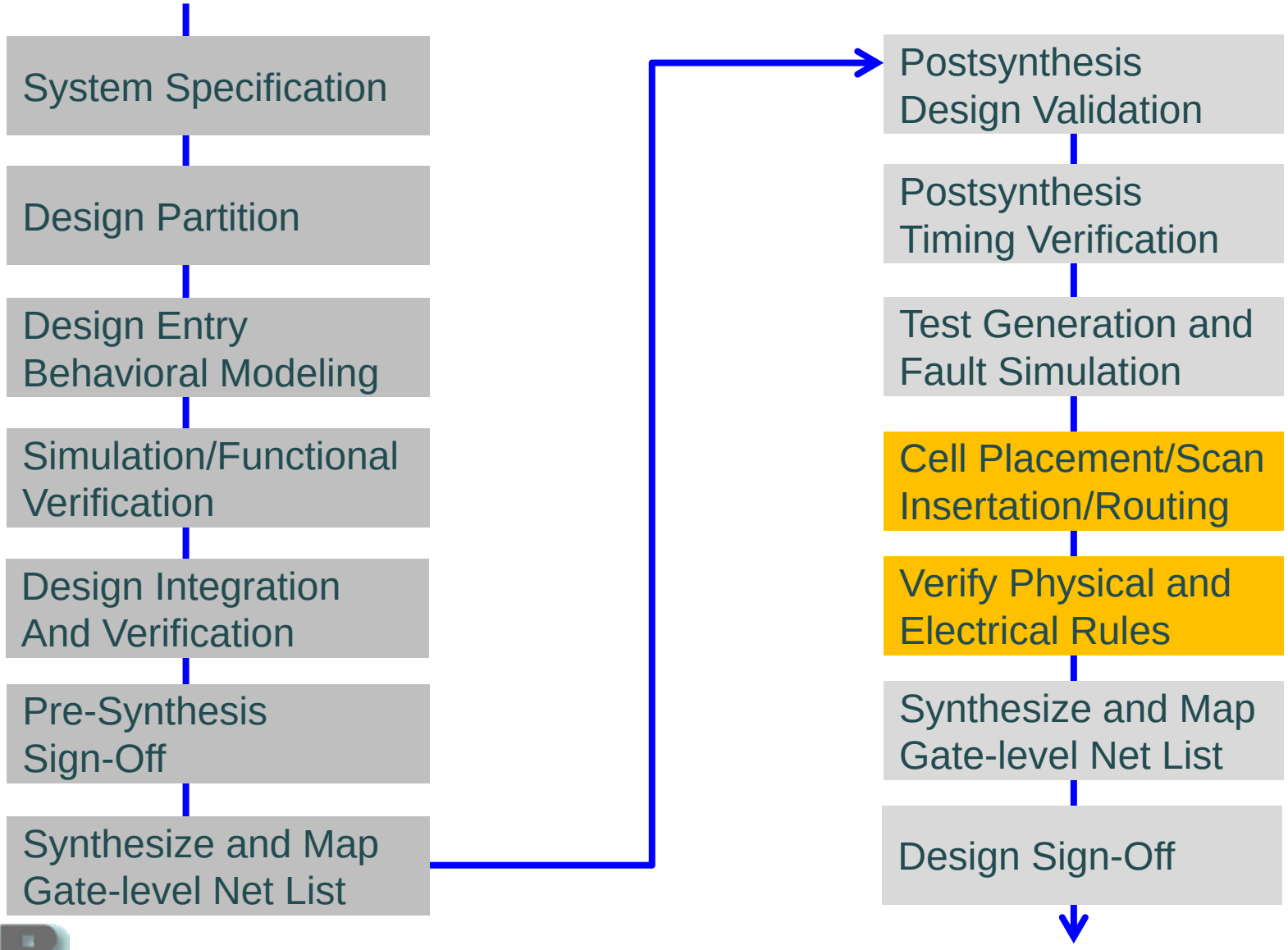
Design Sign-off

- All design constraints have been met
- Timing specifications have been met
- Mask set ready for fabrication

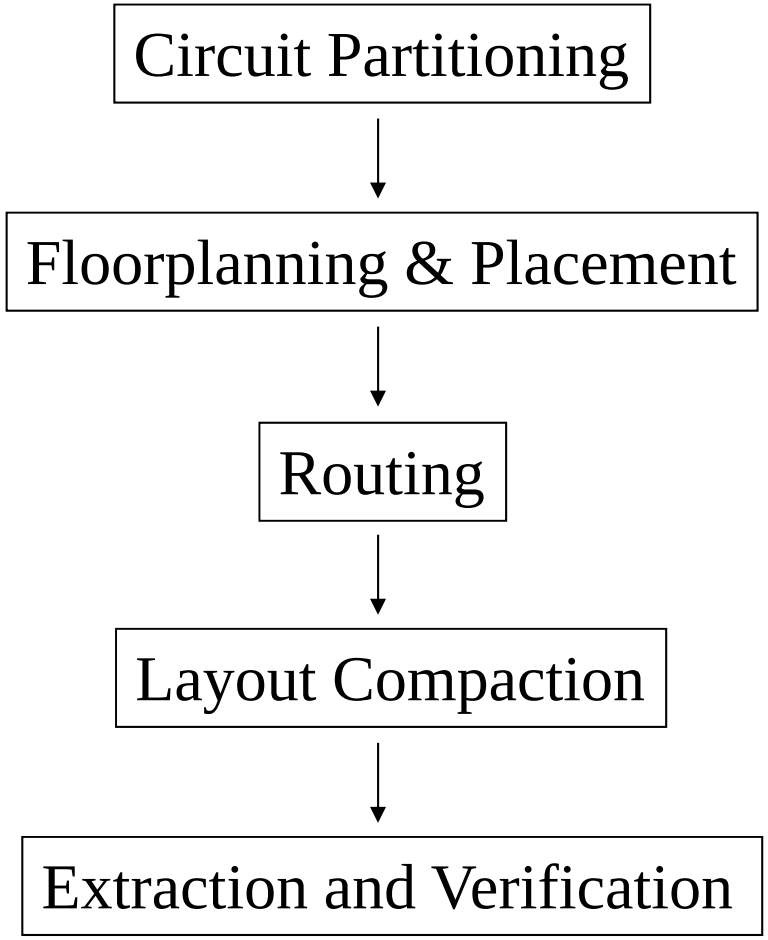
Design Methodology: Big Picture



Design Methodology: Big Picture

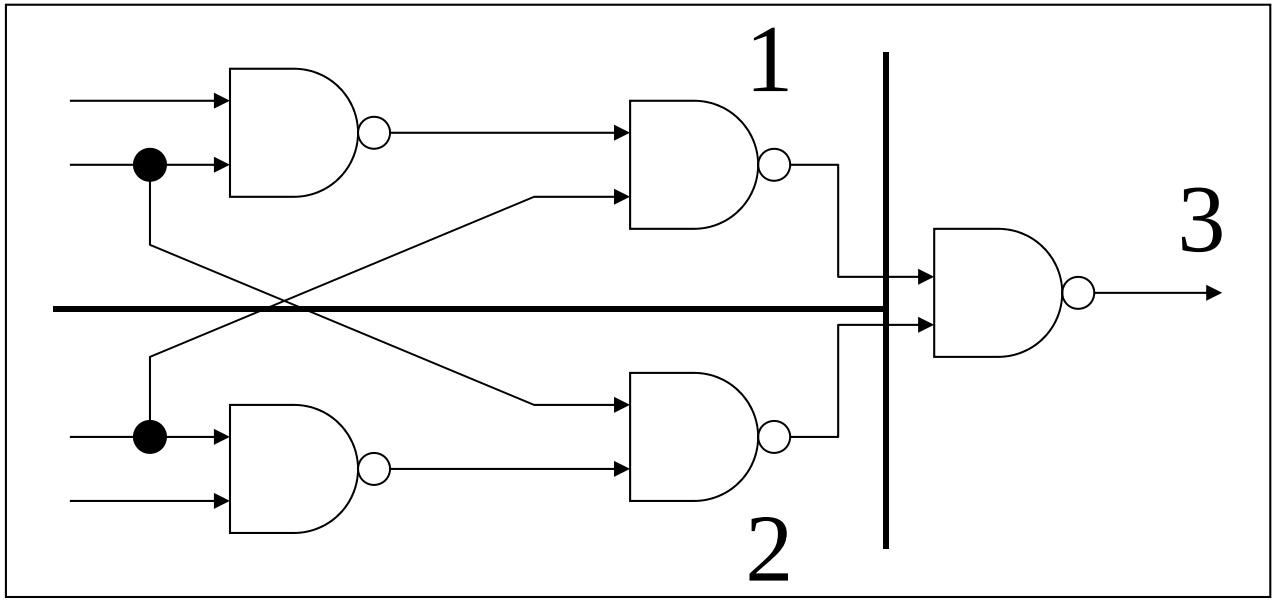


Physical Design Cycle



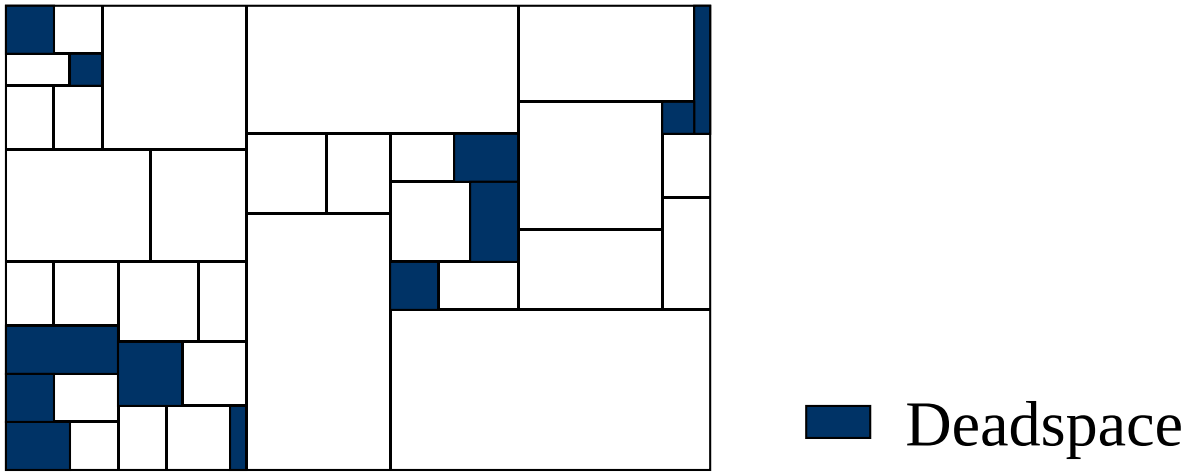
Physical Design Cycle

Circuit Partitioning – Partition a large circuit into sub-circuits (called blocks). Factors like #blocks, interconnections between blocks, … are considered.



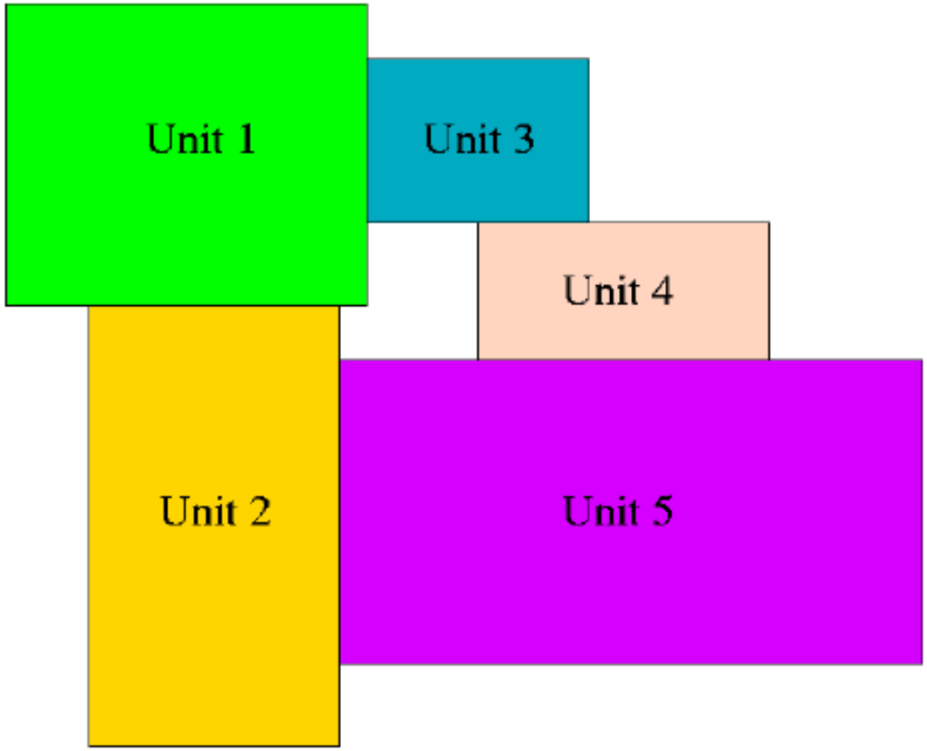
Physical Design Cycle

Floorplanning – Set up a plan for a good layout. Place the blocks at an early stage when details like shape, area, positions of I/O pin, … are not yet fixed.



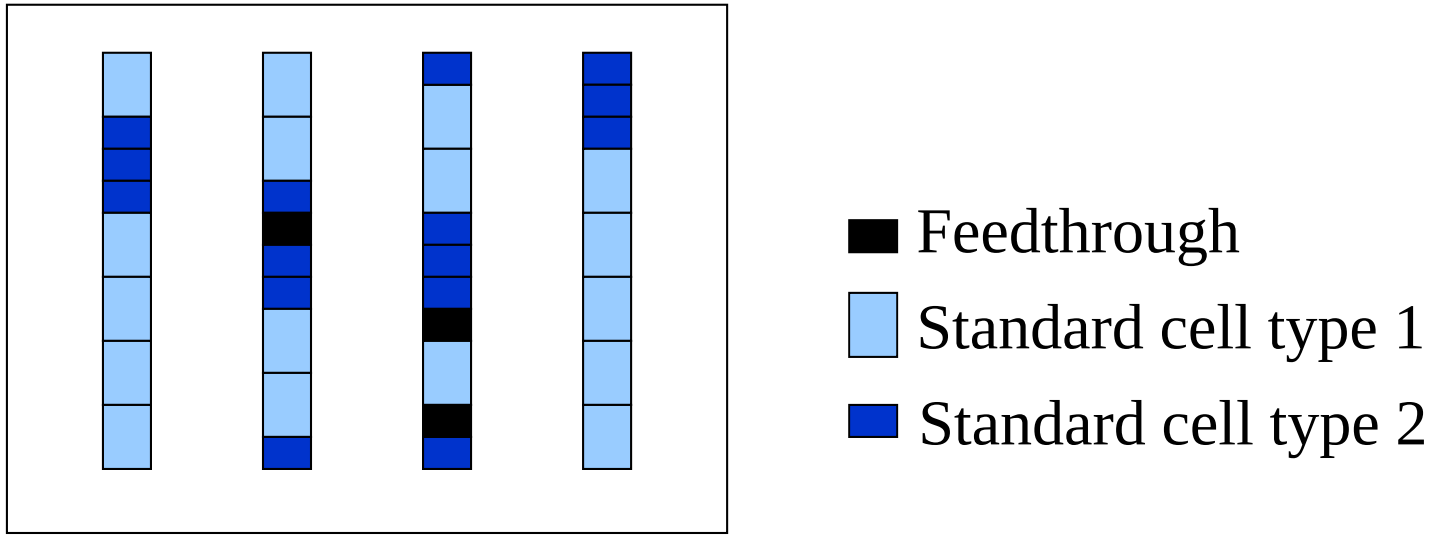
Floorplanning

- Blocks are placed in order to minimize area and the connections between them.



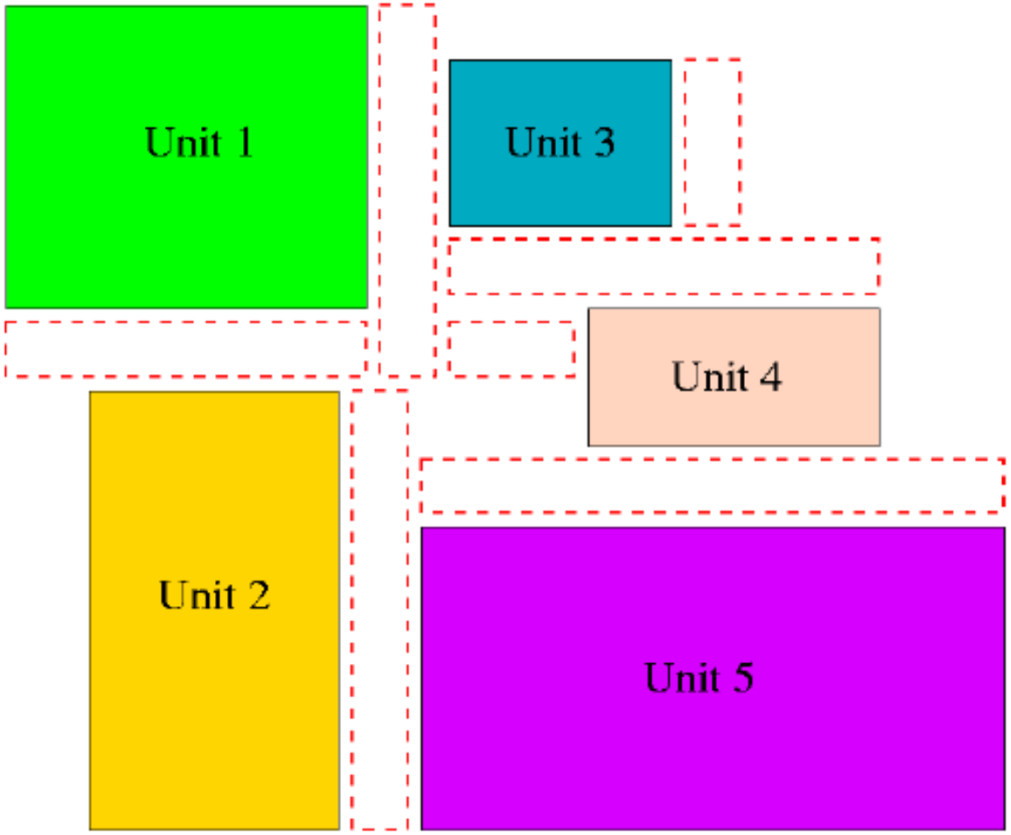
Physical Design Cycle

Placement – Exact placement of the modules (modules can be gates, standard cells, …). Details of the design are known and the goal is to minimize the total area and interconnect cost.



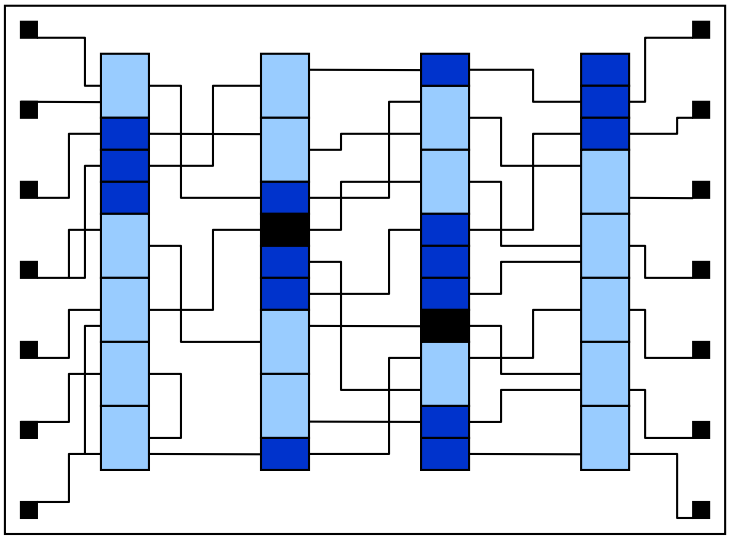
Channel definitions

- Blocks are placed so empty rectangular spaces are left between them. These spaces will be later used to make the interconnection.



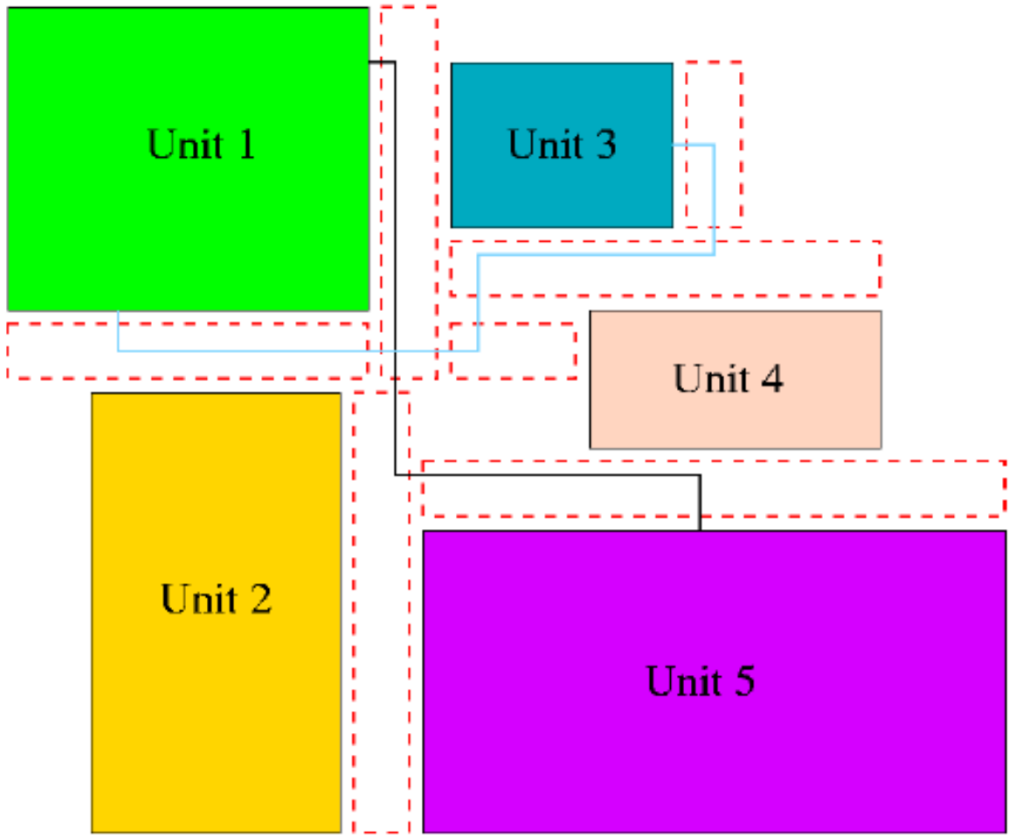
Physical Design Cycle

Routing – Complete the interconnections between the modules. Factors like critical path, wire spacing, … are considered. Include *global routing* and *detailed routing*.



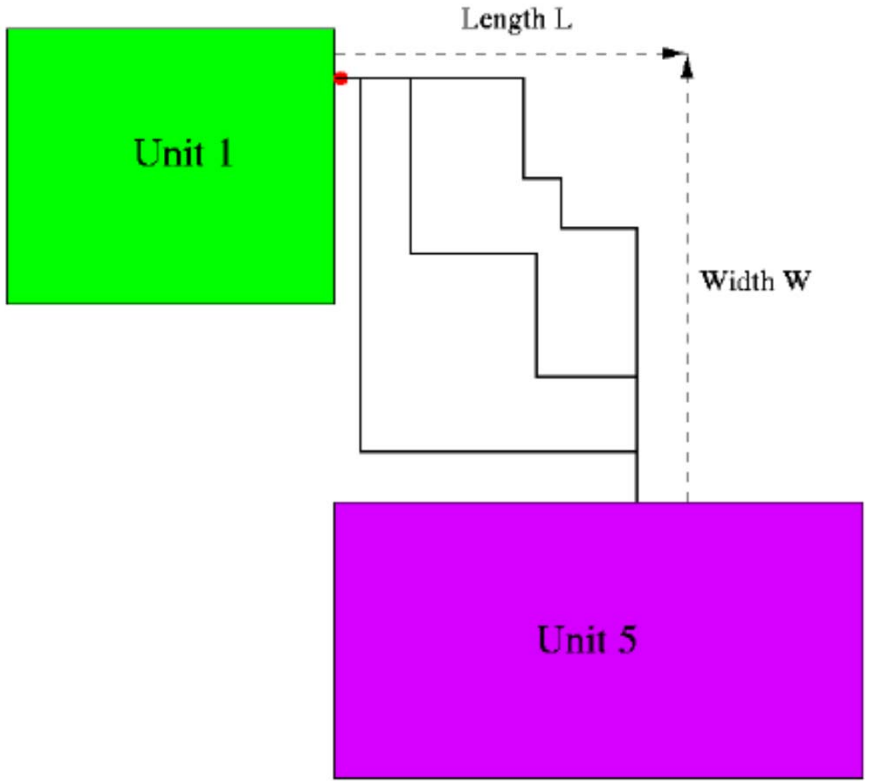
Global Routing

- Each connection will go from each origin block, through the channels until the end block.



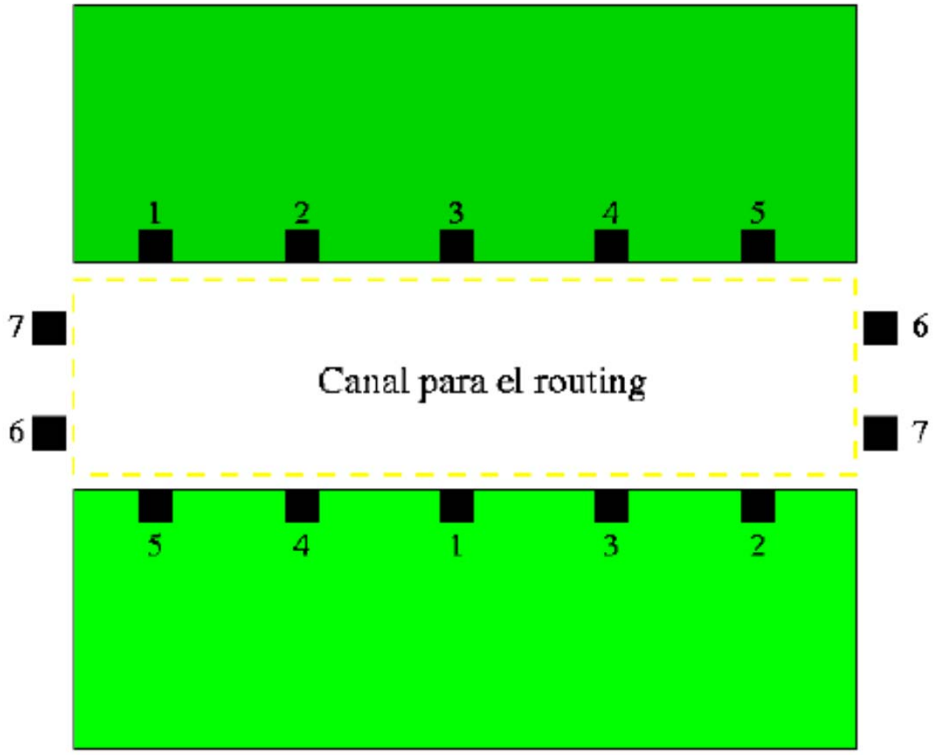
Global Routing

- The length of the connections will depend on the situation of the blocks rather than the way the routing is done.



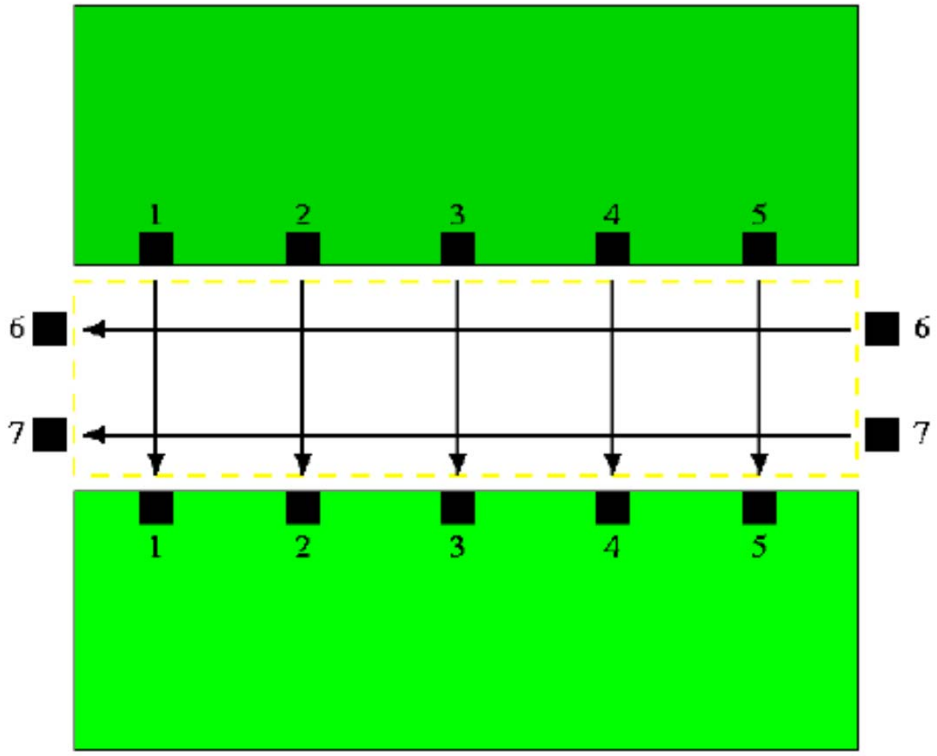
Detailed Routing

- The space required for each channel will depend on the complexity and density of the connections.



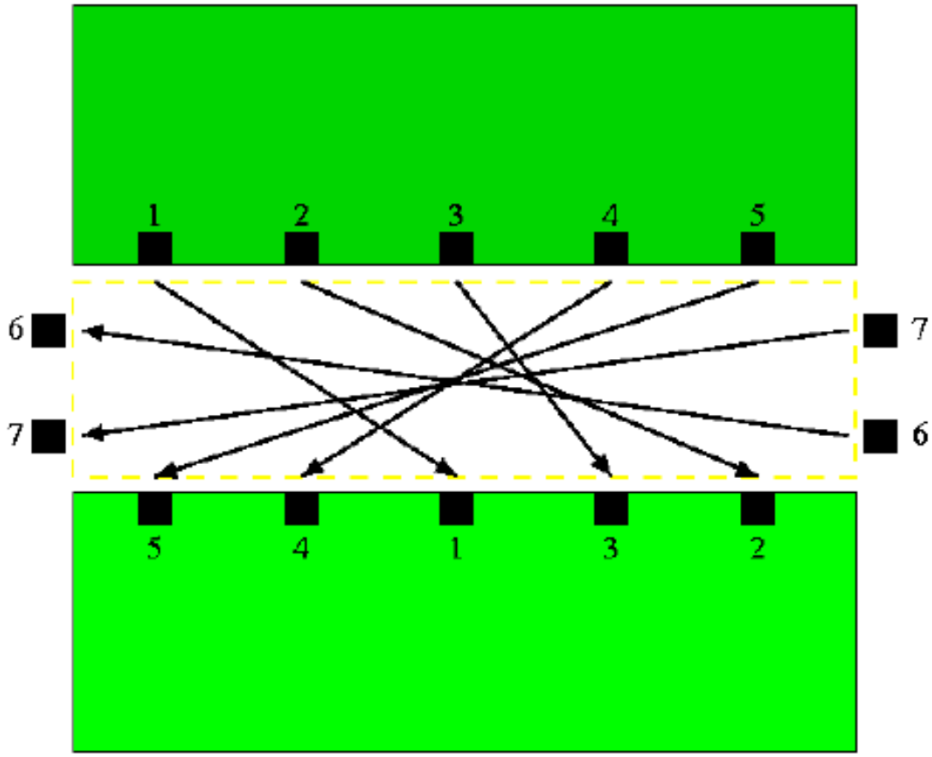
Detailed Routing

- Aligned connections require a smaller channel (similar to that of a bus).



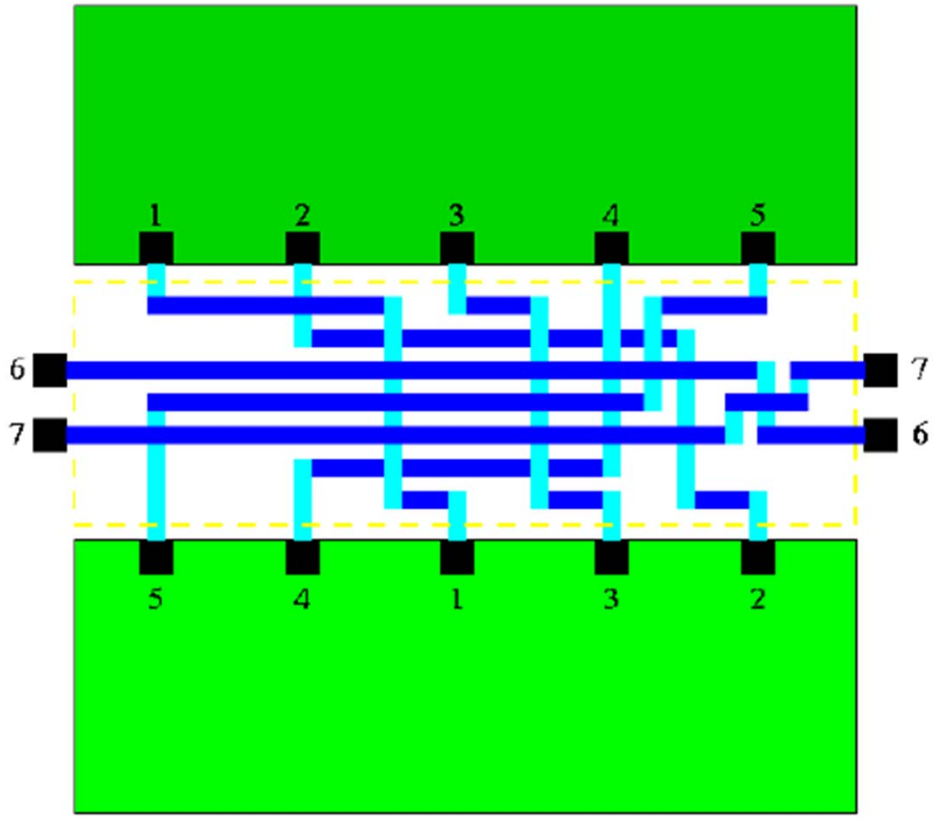
Detailed Routing

- Non-aligned connections increment the complexity of the routing and it increases the amount of space required.



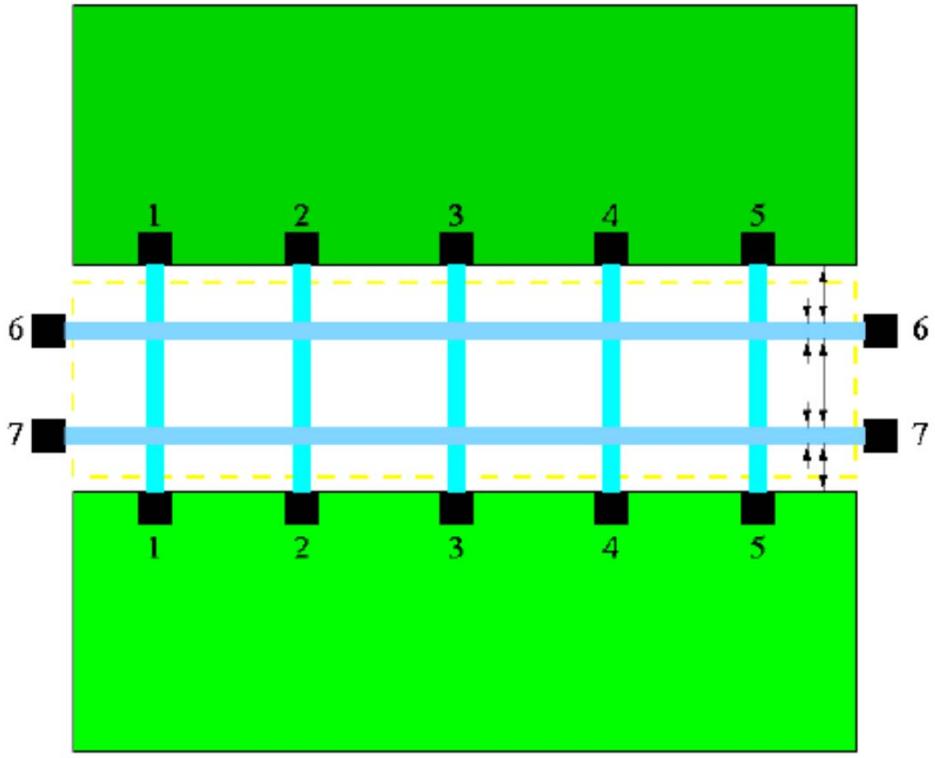
Detailed Routing

- The number of crossings determines the space required. The size of the channel may have to be increased.



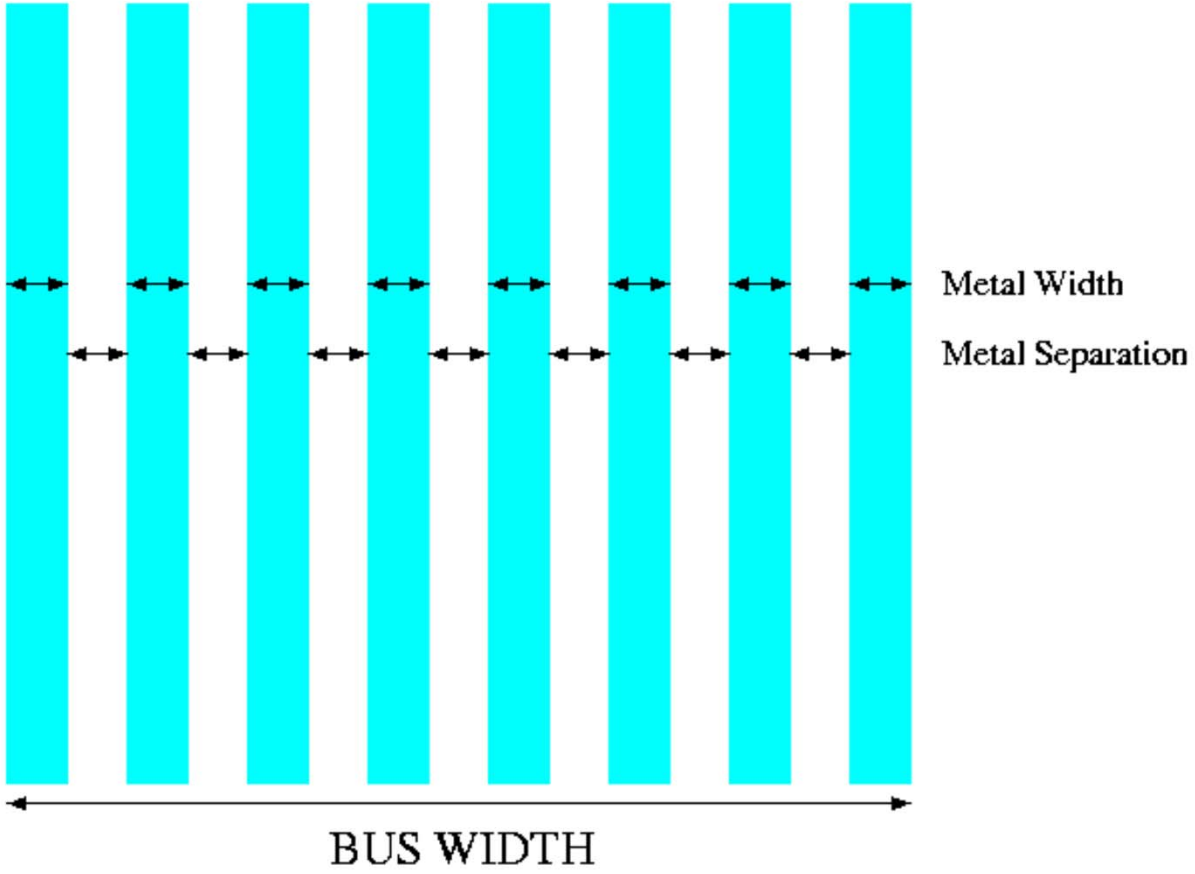
Detailed Routing

- Aligned connections reduce dramaticaly the area required of the channel for completing the routing.



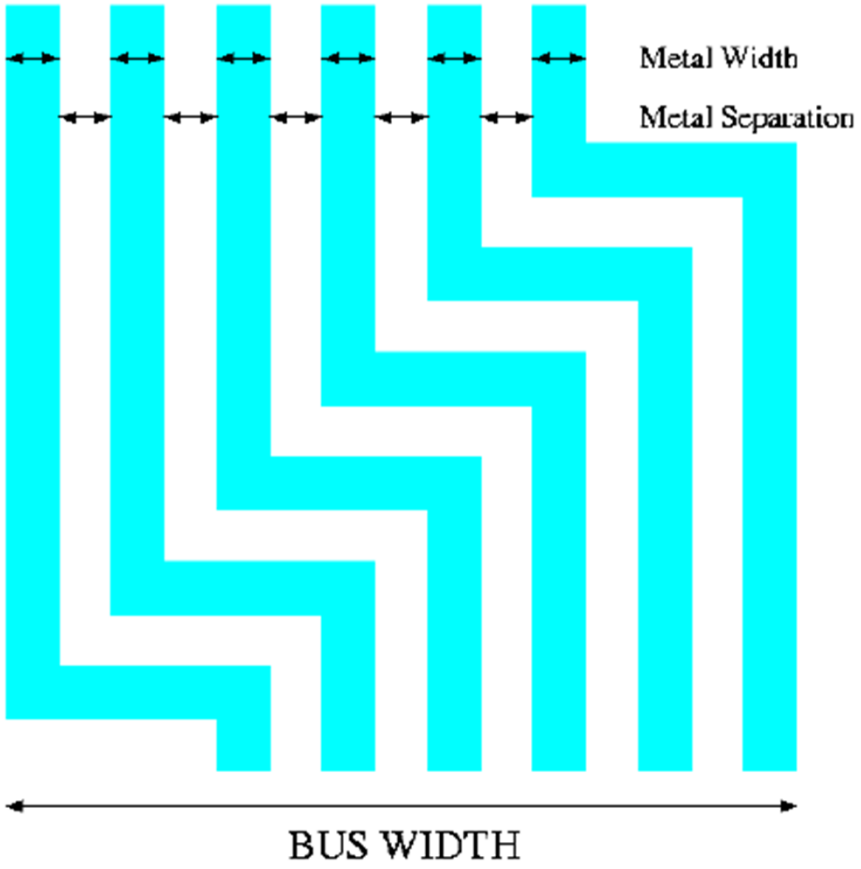
Bus area

- The area of each bus is proportional to the number of bits of the bus.



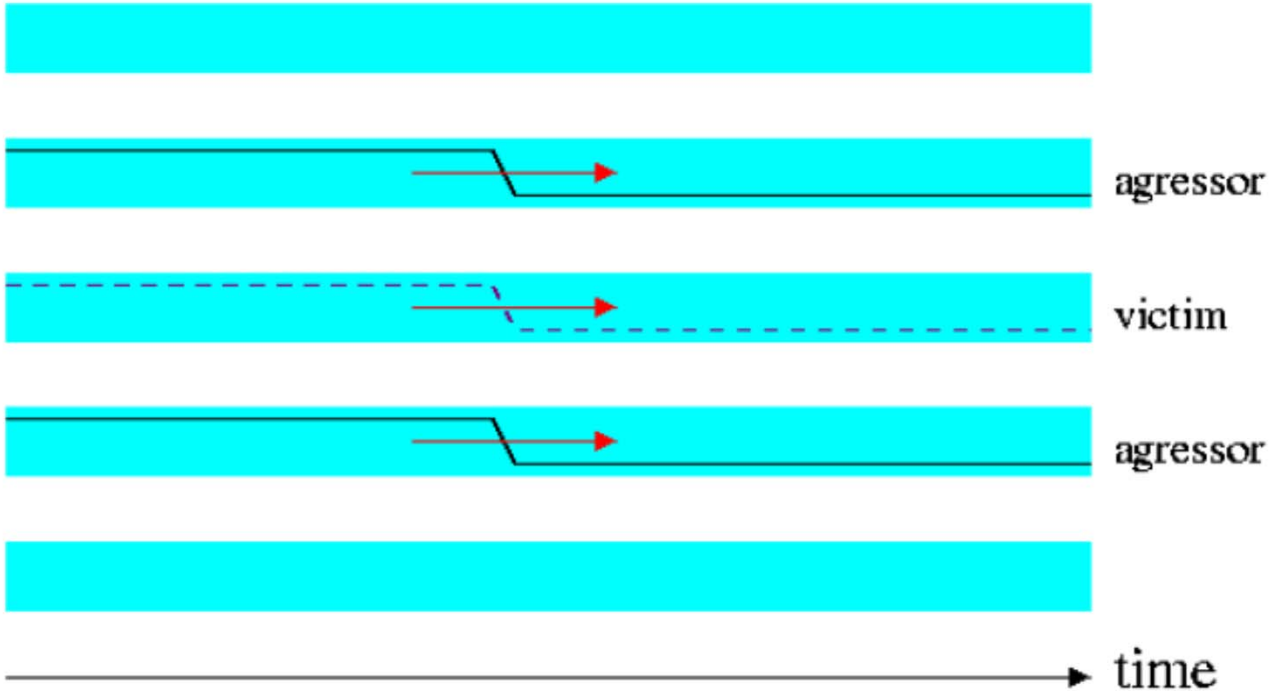
Bus area

- Turns are implemented in the “Manhattan” style. This increases effective area.



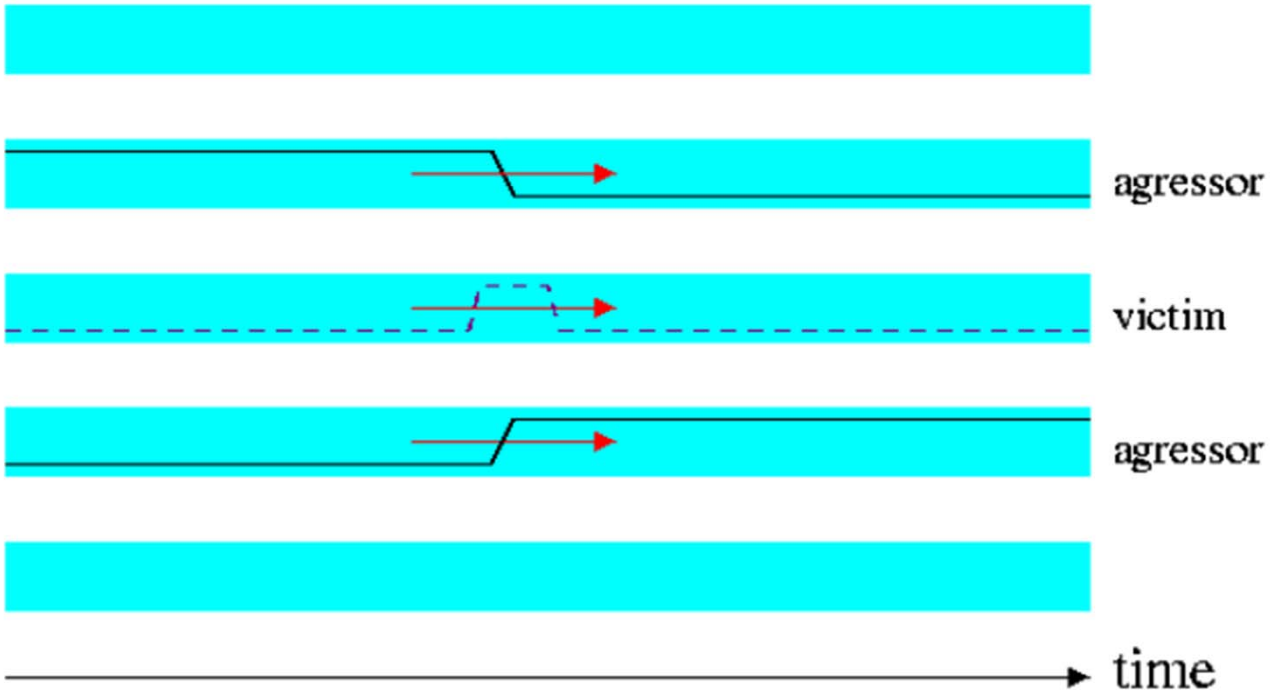
Bus cross-coupling

- The bits in a bus can interfere negatively between them: cross-coupling. (1st example)



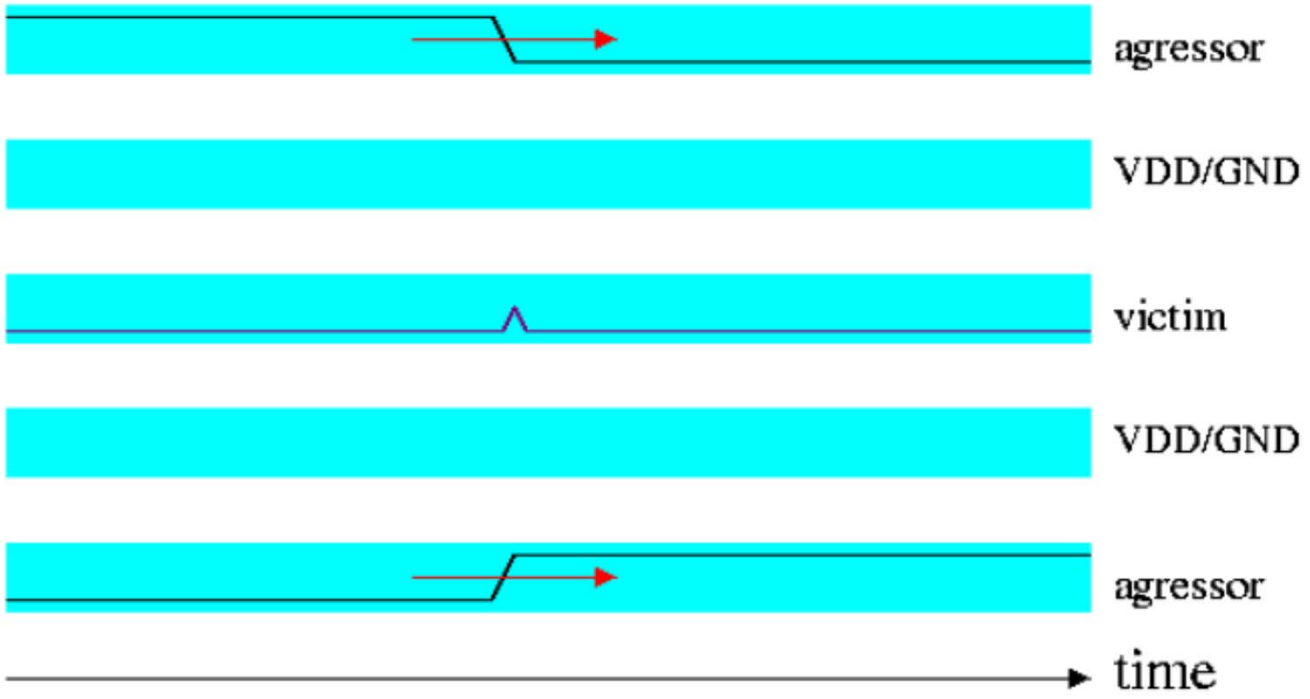
Bus cross-coupling

- The bits in a bus can interfere negatively between them: cross-coupling. (2nd example)



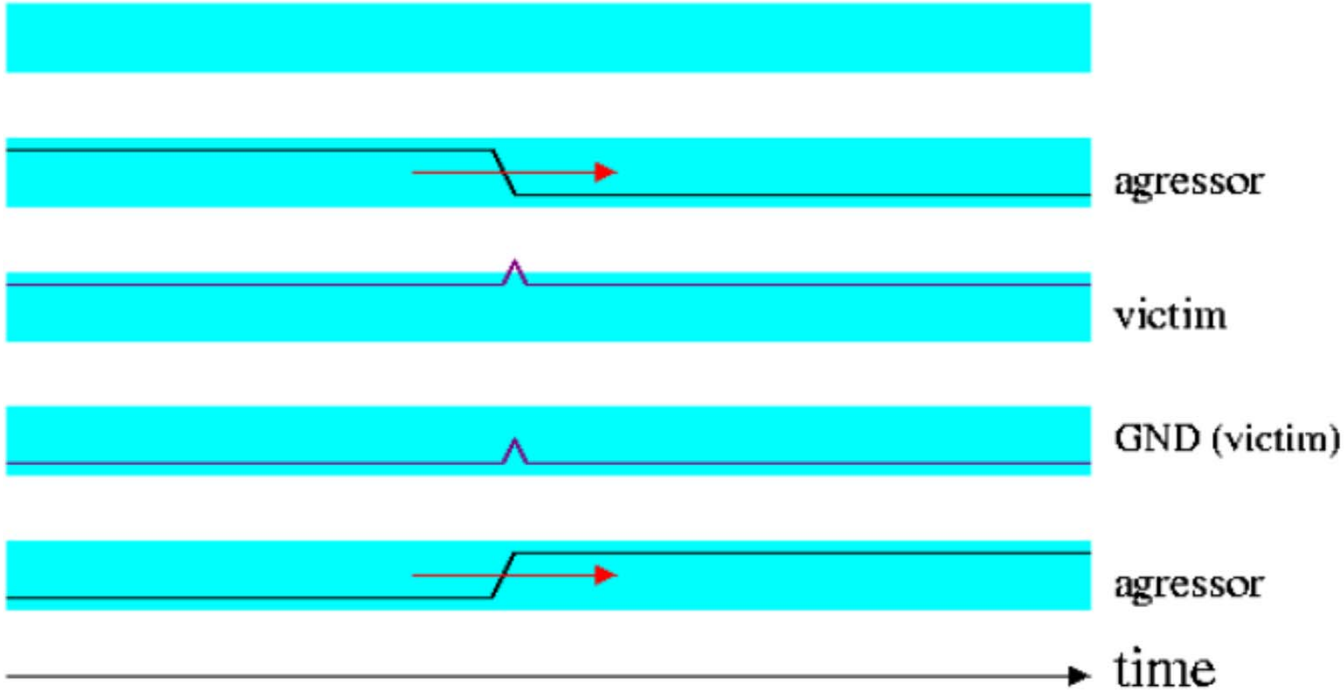
Bus cross-coupling

- If this is the case, the victim must be “protected” using any isolating technique (e.g. increasing the separation).



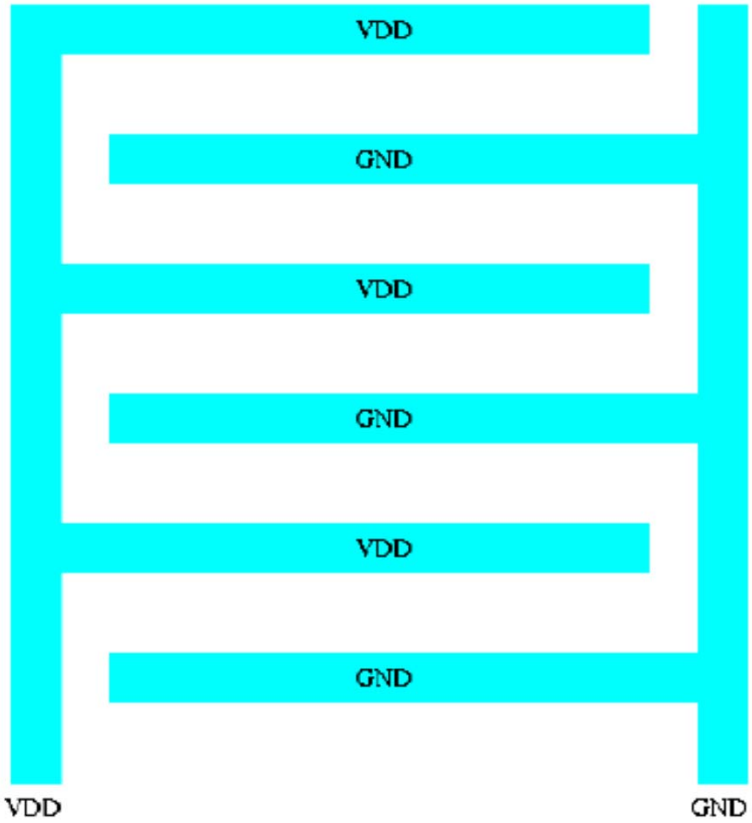
Bus cross-coupling

- Another solution is to use differential information between one signal and the neighbouring GND signal: the glitch appears in both.
- We can measure the difference among both victims.



Power distribution

- Interlazed distribution minimizes the number of metal levels required. The thickness of each channel will vary according to the power consumption.



Physical Design Cycle

Compaction – Compress the layout from all directions to minimize the chip area.

Verification – Check correctness of the layout. Include DRC (*Design Rule Checking*), *circuit extraction* (generate a circuit from the layout to compare with the original netlist), *performance verification* (extract geometric information to compute resistance, capacitance, delay, ...)

Summing up Physical Design

- The area of the blocks is predictable:
  - Depends of the number of transistors and the spacing.
  - It does not depend on the connections.
- The area of the connections is not much predictable.
- If the blocks are larger than the routing: the connections are determined by the separation among the blocks.
- If the routing dominates the blocks (in the same way as buses): The buses behave now like blocks.
- In any other case it is needed to simulate the connections to estimate their area (very sensible to the floorplanning).

Implementation alternatives

Implementation Alternatives

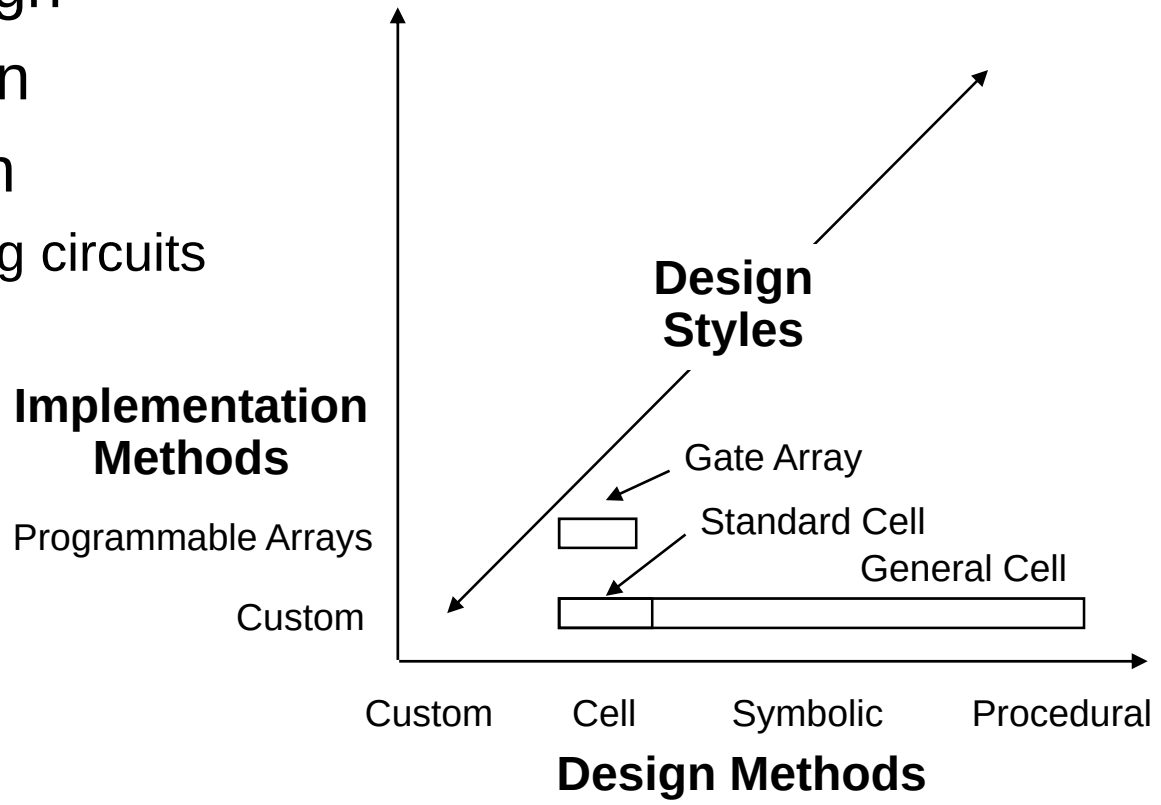
- Alternatives
  - integrated circuits
    - programmable arrays - e.g. ROM, FPGA
    - full custom fabrication
  - hybrid integrated circuits
    - silicon-on-silicon – 3D modules
  - circuit boards
    - discrete wiring - wire-wrap
    - printed circuits
- Design rules
  - topology and geometry constraints
  - imposed by physics and manufacturing
  - example: wires must be > 2 microns wide

IC implementation alternatives

- Full custom design
  - no constraints - output is geometry
    - highest-volume, highest performance designs
  - requires some handcrafted design
    - 5-10 transistors/day for custom layout
  - use to design cells for other methods
  - primary CAD tools
    - layout editor, plotter
- Cell-based design
  - compose design using a library of cells
  - at board-level, cells are chips
  - cell = single gate up to microprocessor
  - primary CAD tools
    - partitioning
    - placement and routing

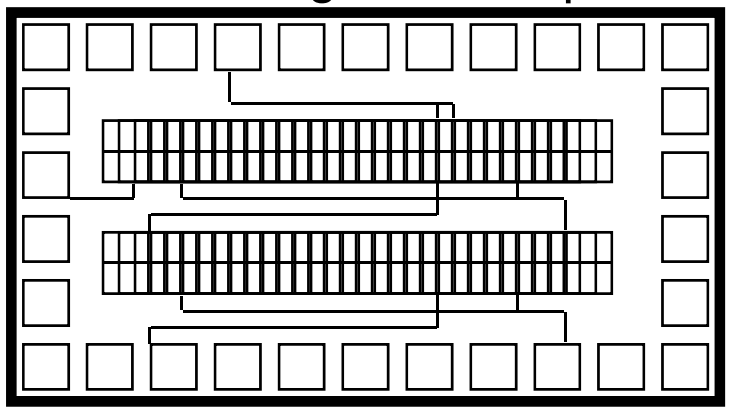
IC implementation alternatives

- Gate array design
  - FPGAs are a form of gate array
- Standard cell design
- General cell design
- Full custom design
  - still used for analog circuits



Gate Array Design

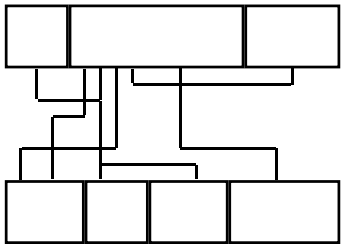
- Array of prefabricated gates/transistors
- Map cell-based design onto gates
- Wire up gates
  - in routing channels between gates
  - over top of gates (sea of gates)
  - predefined wiring patterns to convert transistors to gates
- CAD problems
  - placement of gates/transistors onto fixed sites
  - global and local wire routing in fixed space



FIB

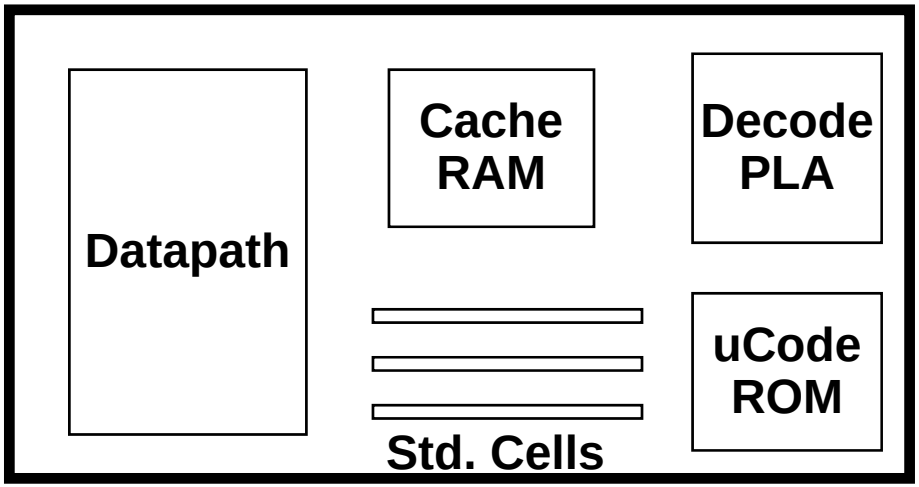
Standard Cell Design

- Design circuit using standard cells
  - cells are small numbers of gates, latches, etc.
- Technology mapping selects cells
- Place and wire them
  - cells placed in rows
    - all cells same height, different widths
  - wiring between rows - channels
- CAD problems
  - cell placement - row and location within row
  - wiring in channels
  - minimize area, delay



General Cell Design

- Generalization of standard cells
- Cells can be large, irregularly shaped
  - standard cells, RAMs, ROMs, datapaths, etc.
- Used in large designs
  - e.g. Pentium has datapaths, RAM, ROM, standard cells, etc.
- CAD problems
  - placement and routing of arbitrary shapes is difficult



Case study

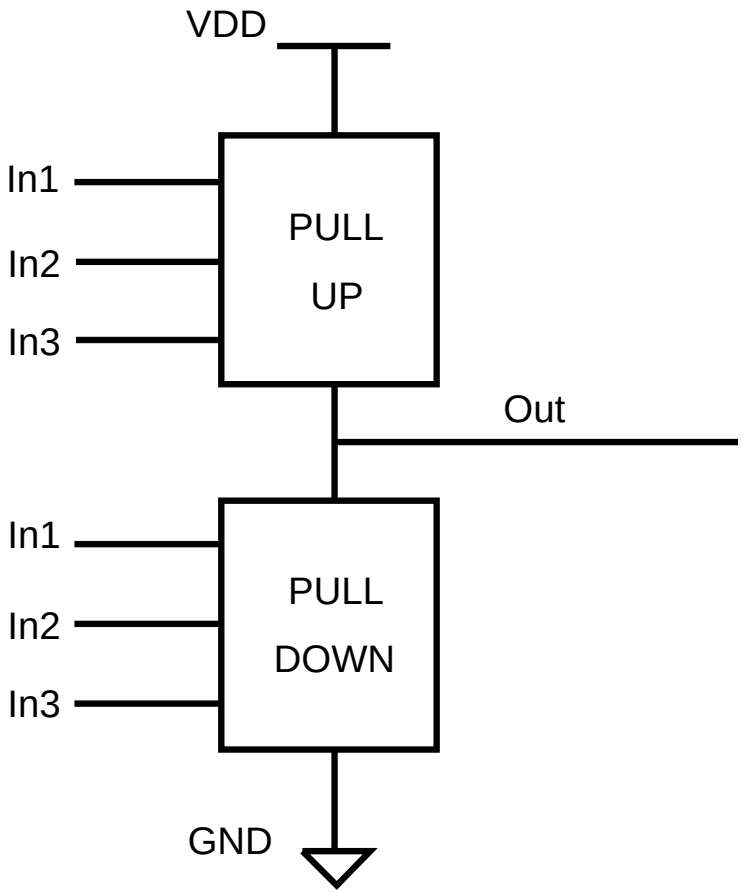
Standard Cell implementation

Standard Cell Implementation

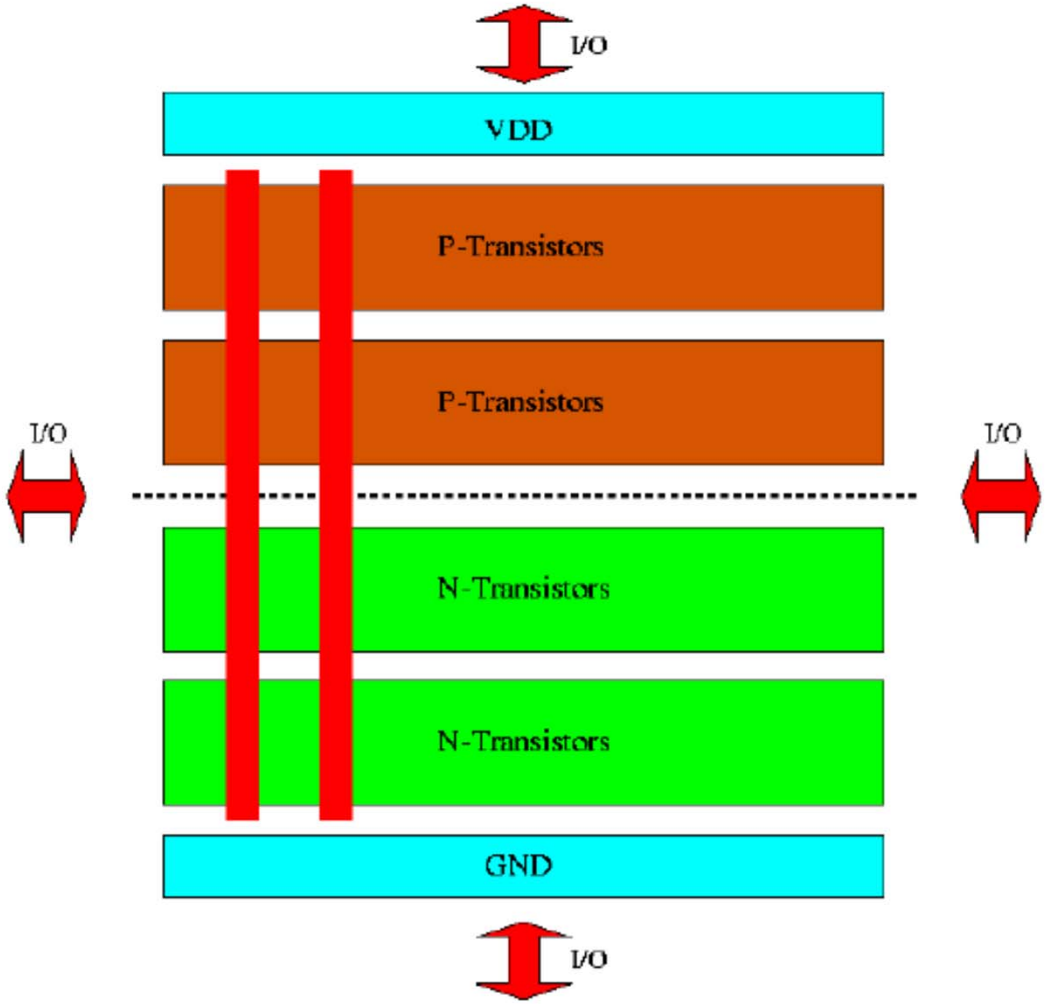
- Cells designed in a certain pattern
- Due to the *standard* pattern they can be used in several circuits

+ Design reuse
- Non-adaptative design

A typical MOS gate

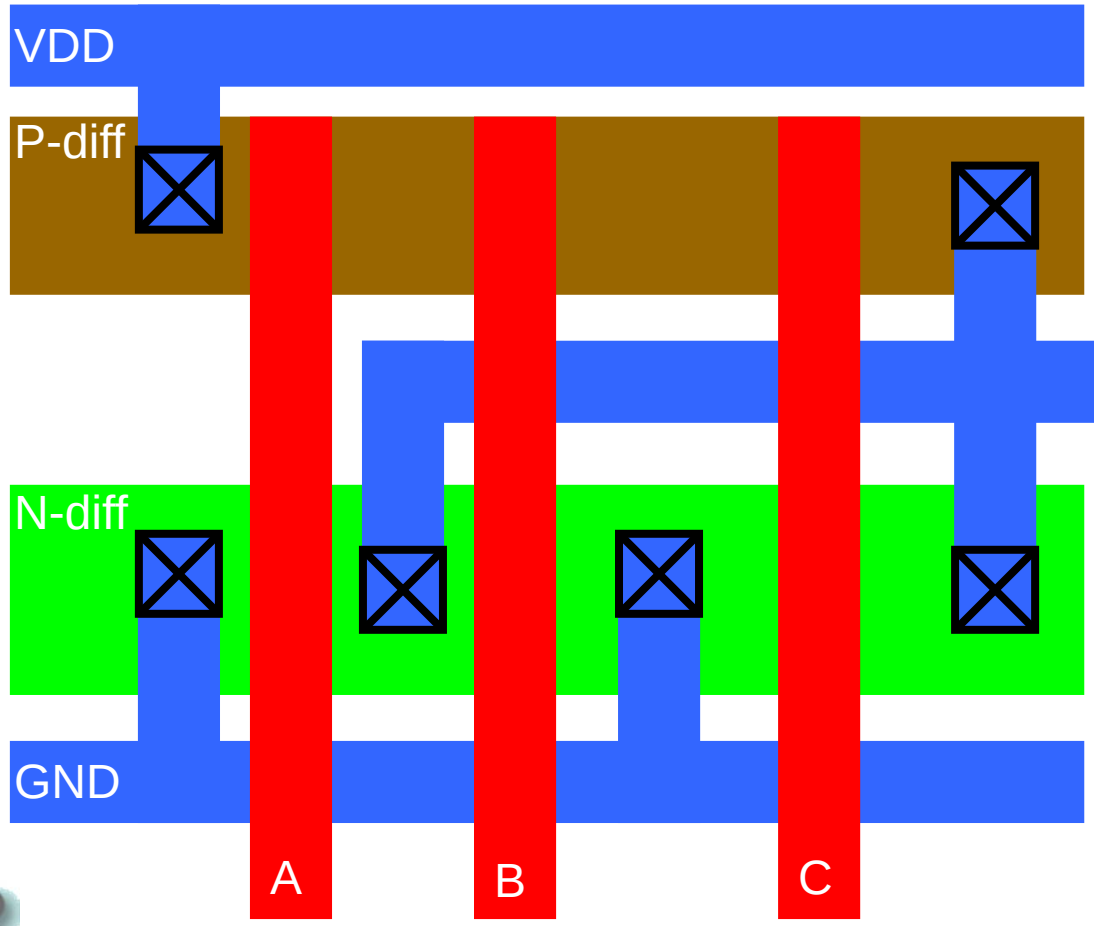


Standard Cell Structure



Standard Cell Structure

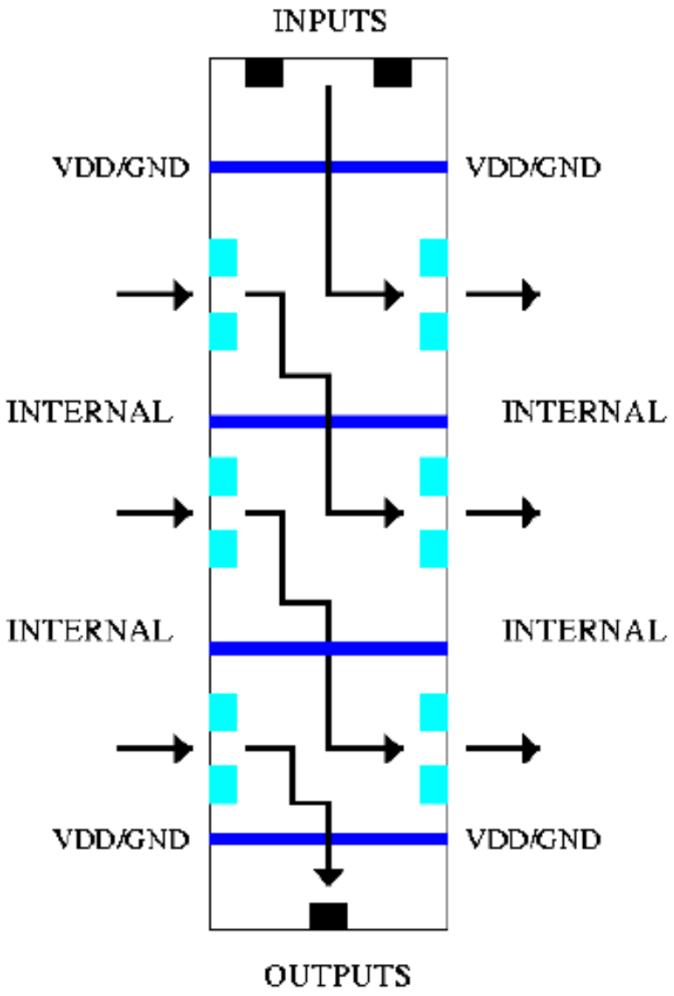
Transistors are placed in a serial way. If we want them in paral·lel we will have to add metal connections.



$$OUT = \overline{A + B + C}$$

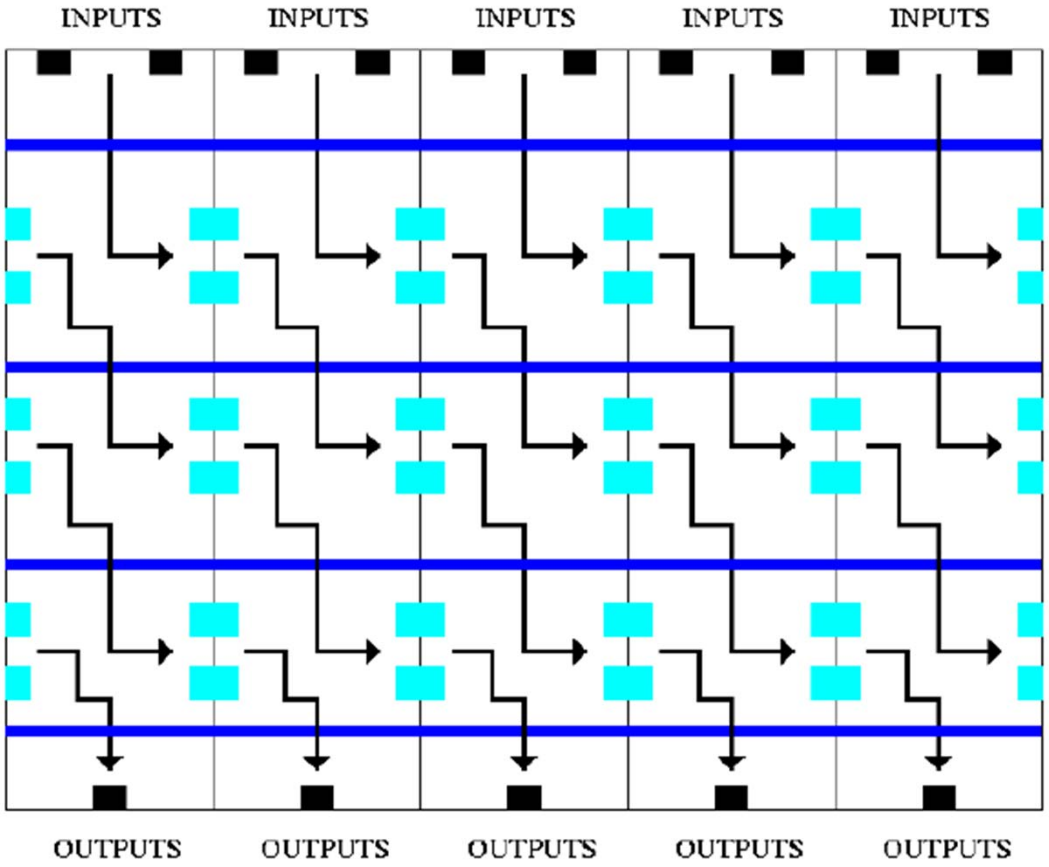
Bit slice design

- We can design a block that implements all the operations for one bit (e.g. in a multi-bit design, as in an adder)
- We have to take into account all input, ouput and internal connections.
- Putting each cell together generates a regular and dense structure.
- Usual way of designing a processor's datapath (registers + FU + shifters).



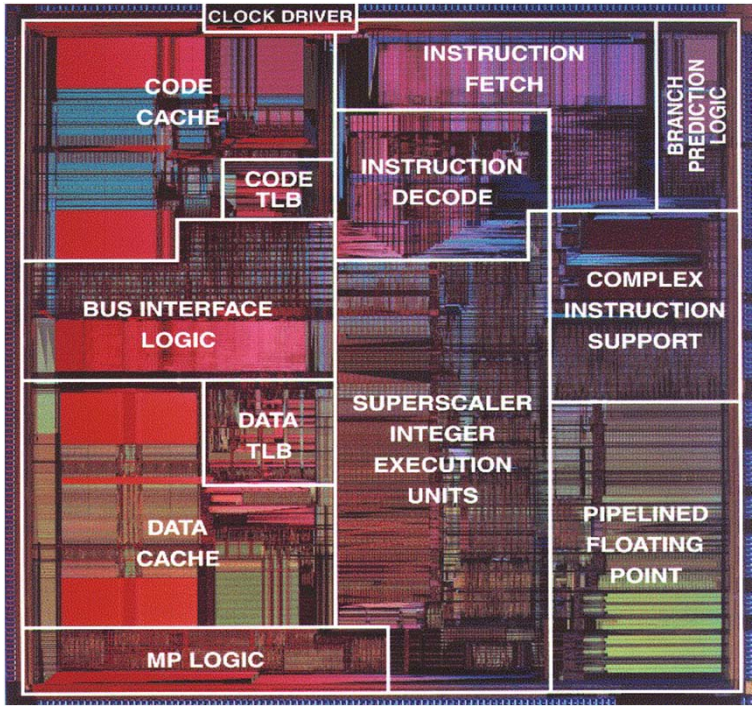
Bit slice design

- Block design from standard 1-bit cells

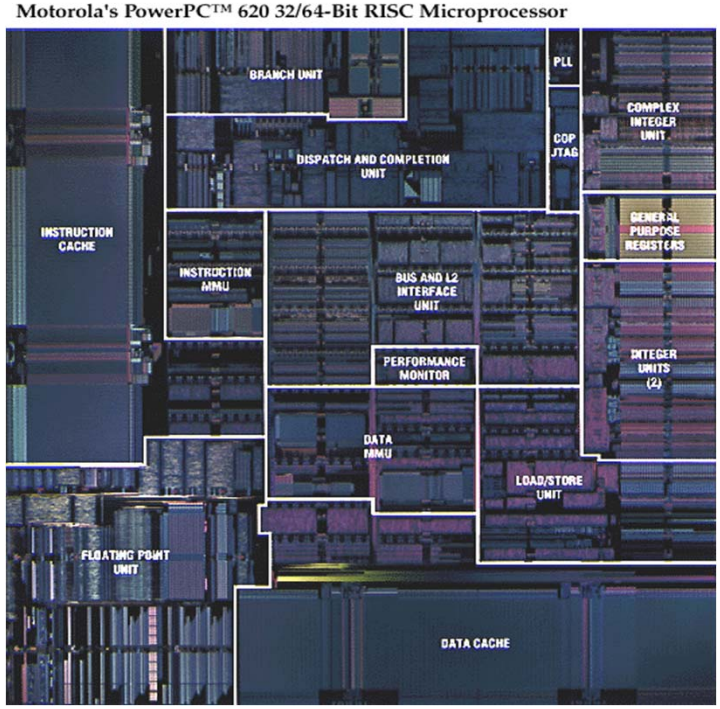


High performance devices

- Mixture of full custom, standard cells and macro's
- Full custom for special blocks: Adder (data path), etc.
- Macro's for standard blocks: RAM, ROM, etc.
- Standard cells for non critical digital blocks



Pentium



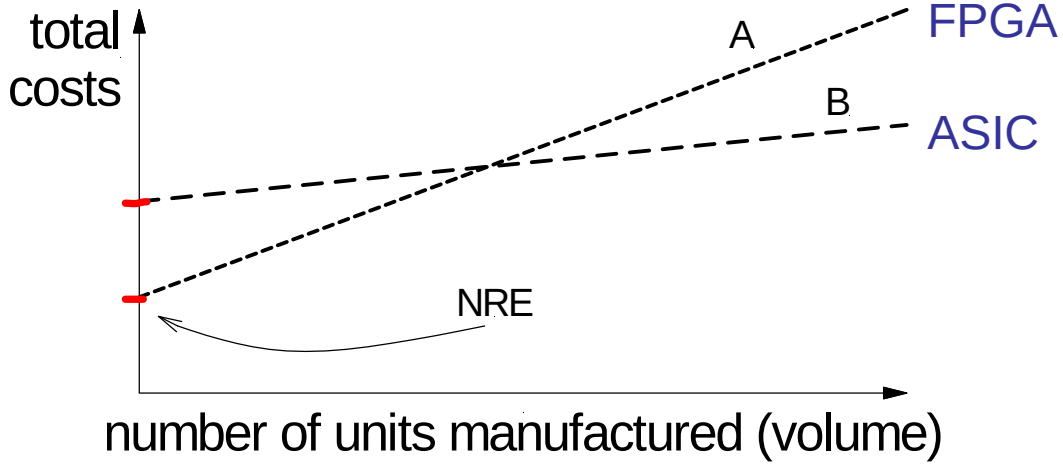
Power PC

Case study

FPGA just for prototyping?

Why FPGAs?

- Custom ICs where sometimes designed to replace the large amount of glue logic:
  - reduced system complexity and manufacturing cost, improved performance.
  - However, custom ICs are relatively very expensive to develop, and delay introduction of product to market (time to market) because of increased design time.
- Note: need to worry about two kinds of costs:
  1. cost of development, sometimes called non-recurring engineering (NRE)
  2. cost of manufacture
  - A tradeoff usually exists between NRE cost and manufacturing costs

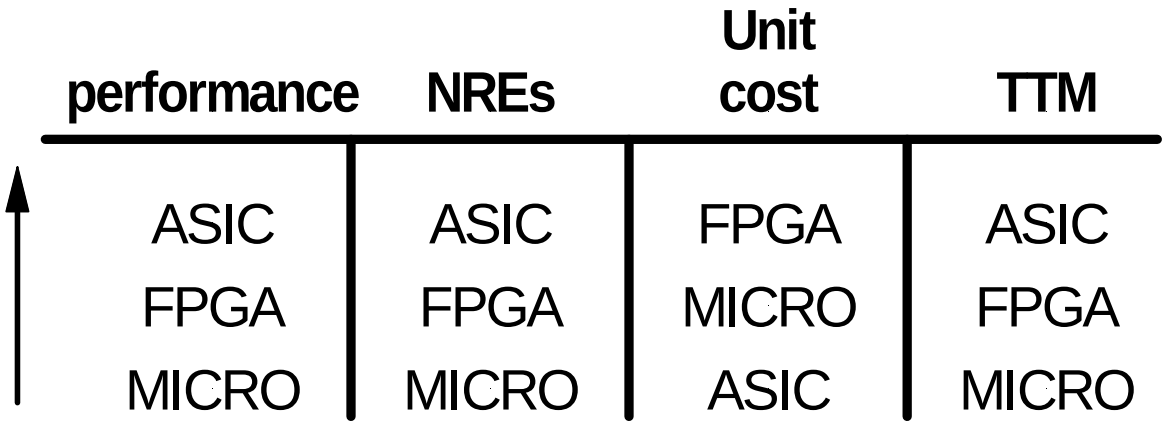


Why FPGAs?

- Therefore the custom IC approach was only viable for products with very high volume (where NRE could be amortized), and which were not time to market (TTM) sensitive.
- FPGAs were introduced as an alternative to custom ICs for implementing glue logic:
  - improved density relative to discrete SSI/MSI components (within around 10x of custom ICs)
  - with the aid of computer aided design (CAD) tools circuits could be implemented in a short amount of time (no physical layout process, no mask making, no IC manufacturing), relative to ASICs.
    - lowers NREs
    - shortens TTM
- Because of Moore's law, the density (gates/area) of FPGAs continued to grow through 00's to the point where major data processing functions can be implemented on a single FPGA.

Why FPGAs?

- FPGAs continue to compete with custom ICs for special processing functions (and glue logic) but now also compete with microprocessors in dedicated and embedded applications.
  - Performance advantage over microprocessors because circuits can be customized for the task at hand. Microprocessors must provide special functions in software (many cycles).
- Summary:



ASIC = custom IC, MICRO = microprocessor

- Newer FPGAs even combine microprocessor cores, special multiplier circuits, memory blocks, and configurable logic on a single chip.

Summary

- Logic design process influenced by available technology AND economic drivers
  - Volume, Time to Market, Costs, Power
- FPGA offer a valuable new sweet spot
  - Low TTM, medium cost, tremendous flexibility (during and after design is done - field upgrades are possible).
- Fundamentally tied to powerful CAD tools
- Build everything (simple or complex) from one set of building blocks
  - LUTs + FF + routing + storage + IOs