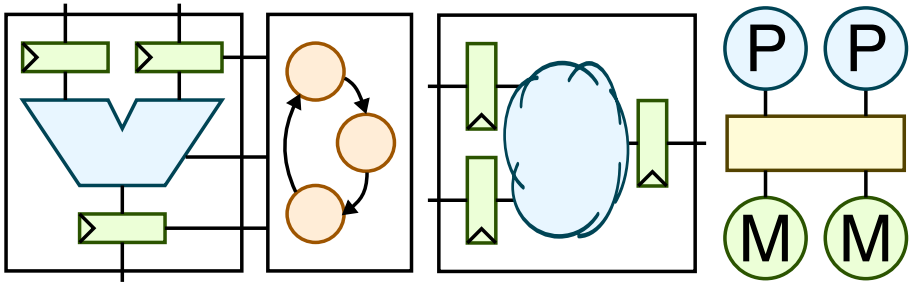
ECE 5745 Complex Digital ASIC Design
Topic 4: Full-Custom Design Methodology

Christopher Batten

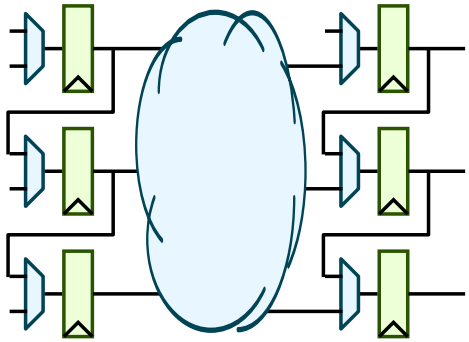
School of Electrical and Computer Engineering
Cornell University

http://www.csl.cornell.edu/courses/ece5745

Part 1: ASIC Design Overview



Topic 1
Hardware Description Languages

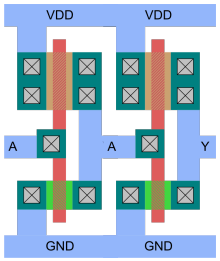


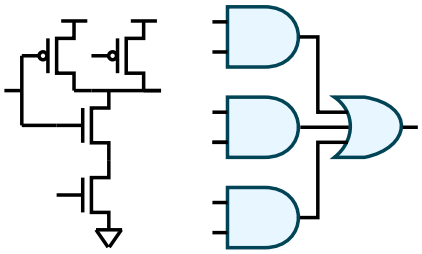
Topic 8
Testing and Verification

Topic 4
Full-Custom Design Methodology

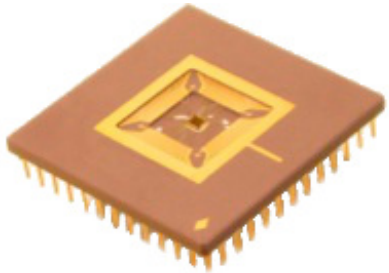
Topic 6
Closing the Gap

Topic 5
Automated Design Methodologies

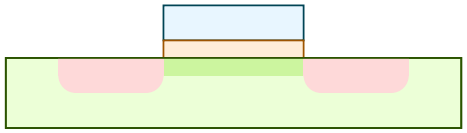




Topic 3
CMOS Circuits



Topic 7
Clocking, Power Distribution, Packaging, and I/O



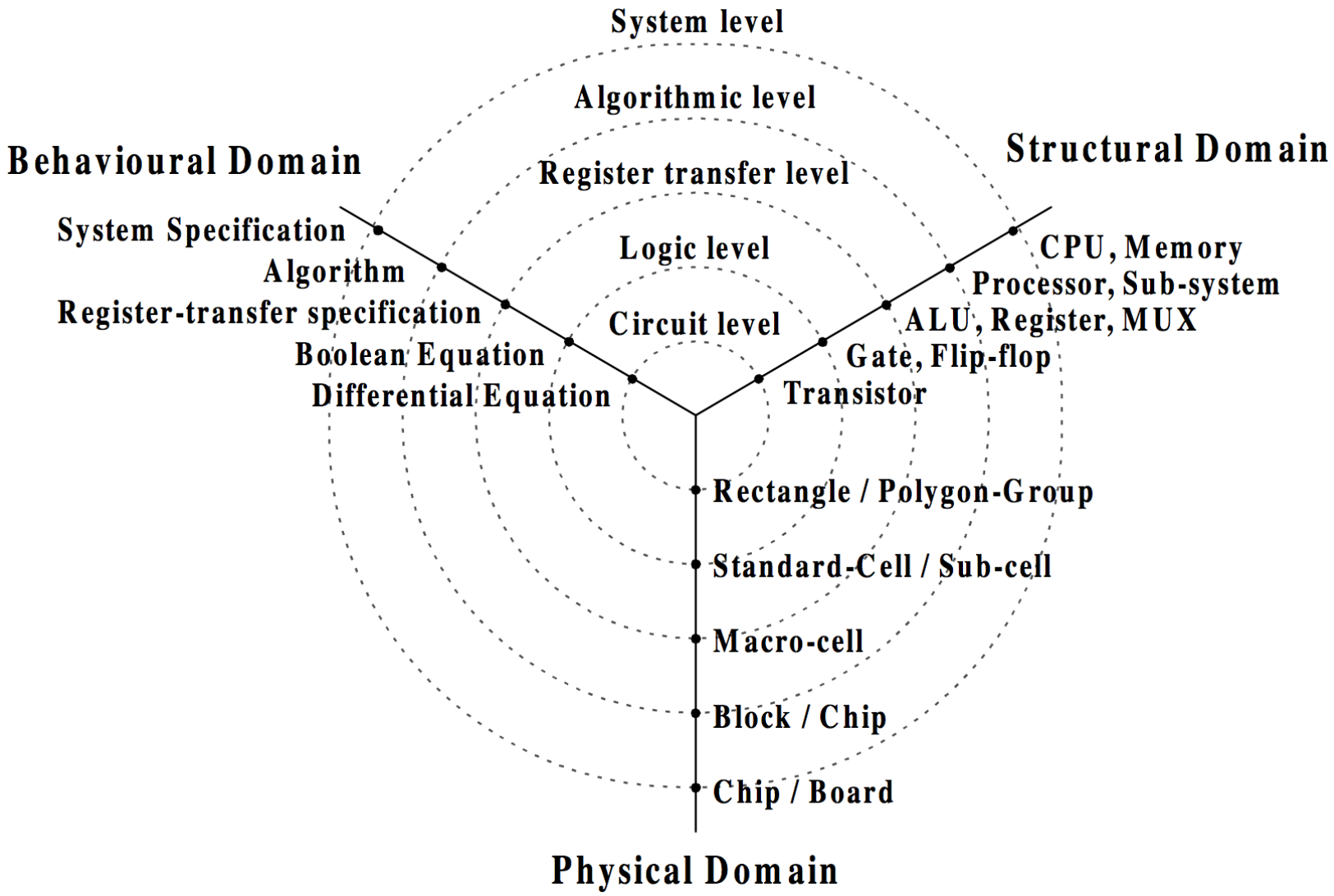
Topic 2
CMOS Devices

Agenda

Design Domains, Abstractions, and Principles
- Modularity
- Hierarchy
- Encapsulation
- Regularity
- Extensibility

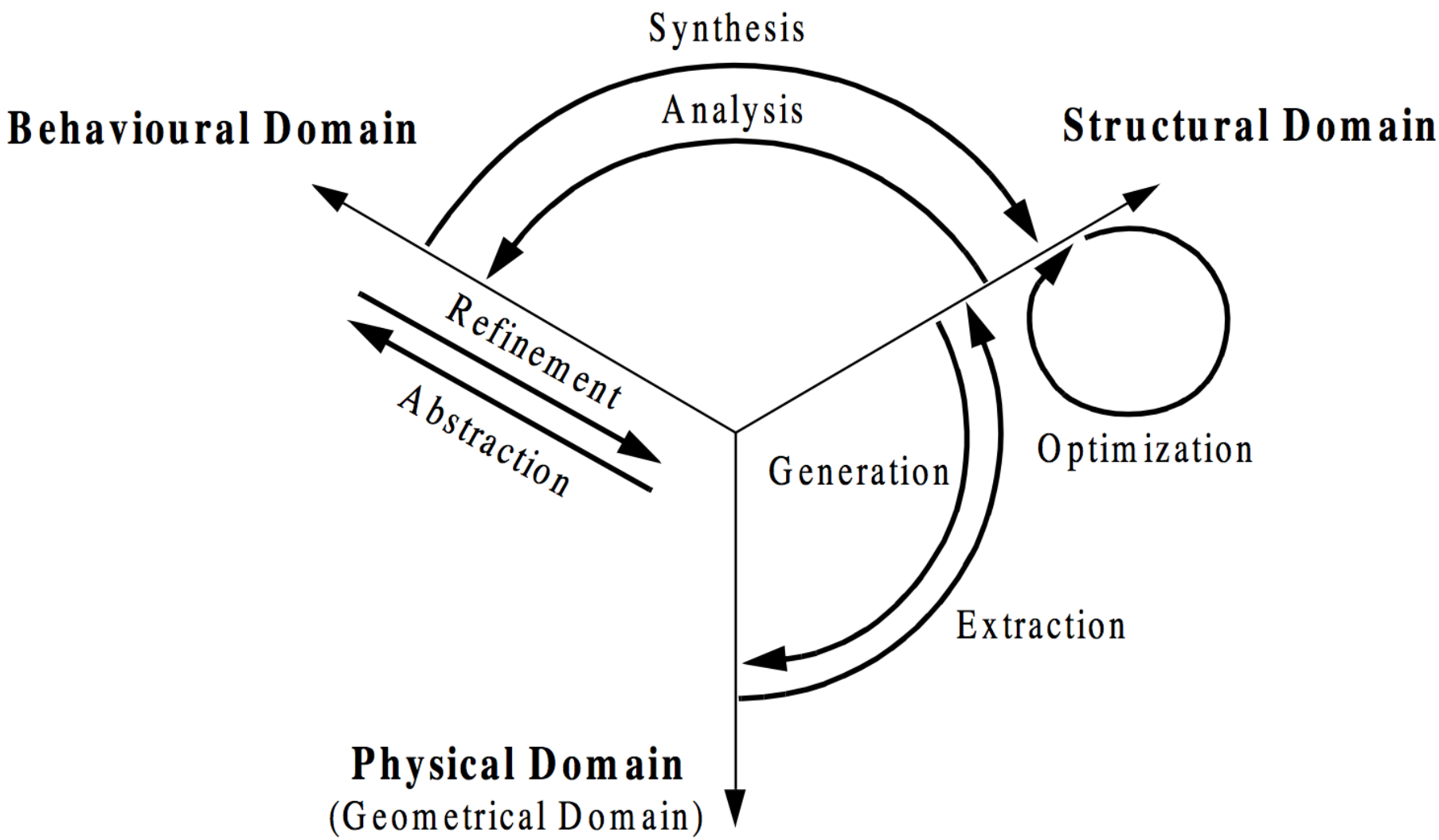
Full-Custom Design

Behavioral, Structural, and Physical Abstractions



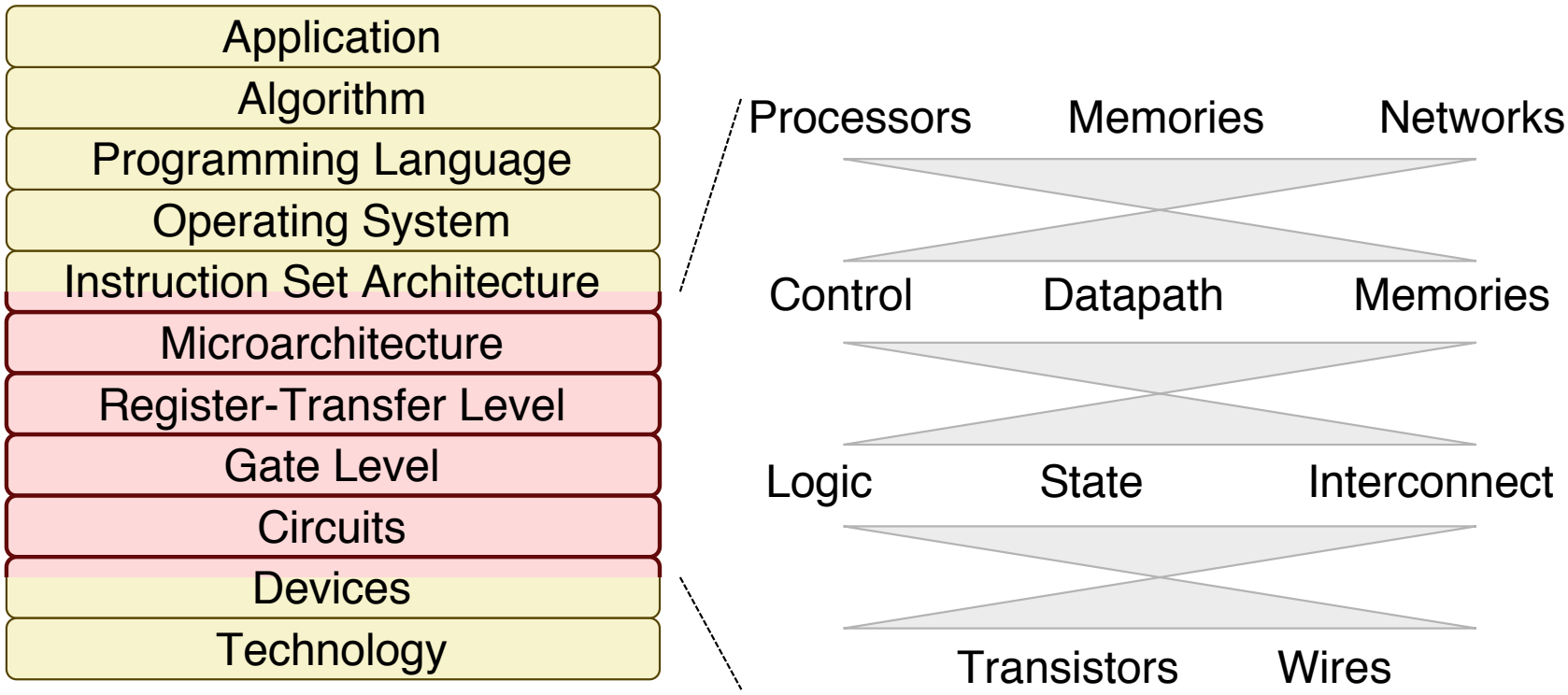
Adapted from [Ellervee'04]

Behavioral, Structural, and Physical Abstractions



Adapted from [Ellervee'04]

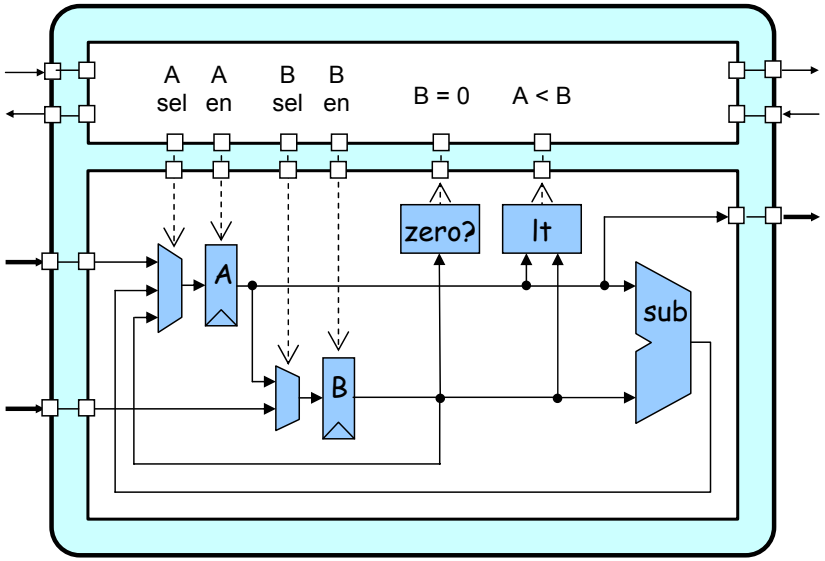
Computer Engineering Stack Abstractions

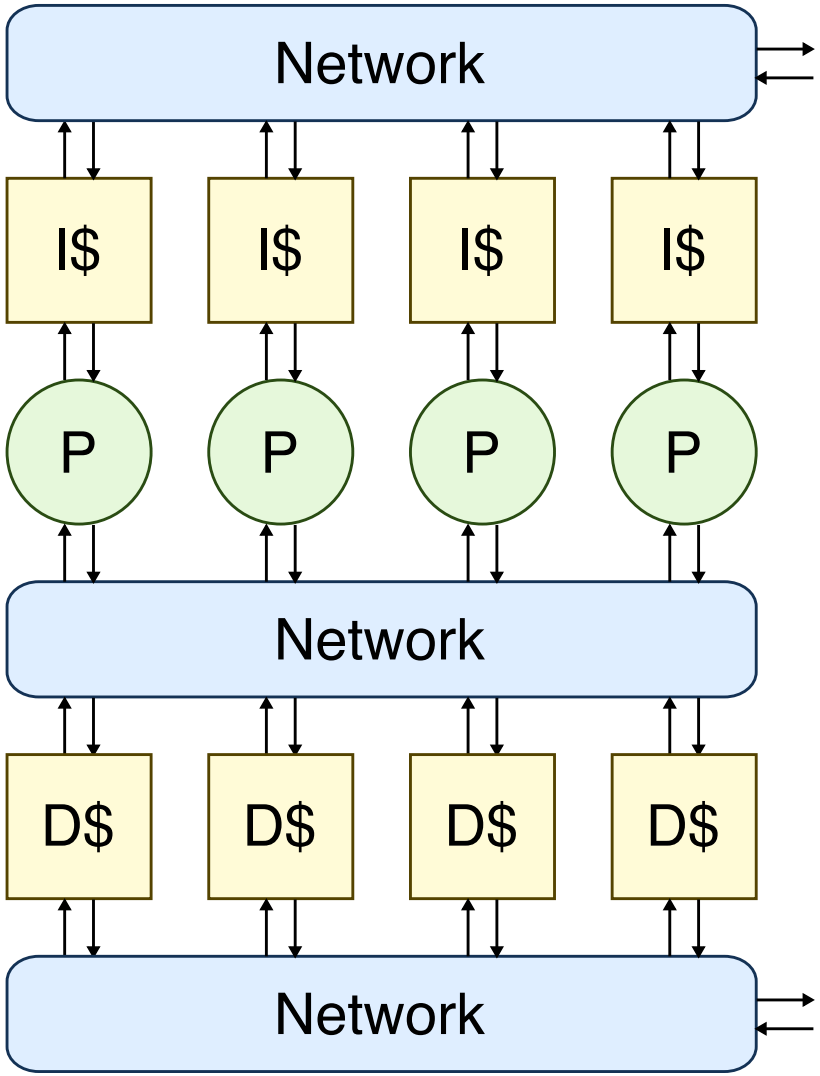


Design Principles in VLSI Design

- Modularity – Decompose into components with well-defined interfaces
- Hierarchy – Recursively apply modularity principle
- Encapsulation – Hide implementation details from interfaces
- Regularity – Leverage structure at various levels of abstraction
- Extensibility – Include mechanisms/hooks to simplify future changes

Design Principle: Modularity



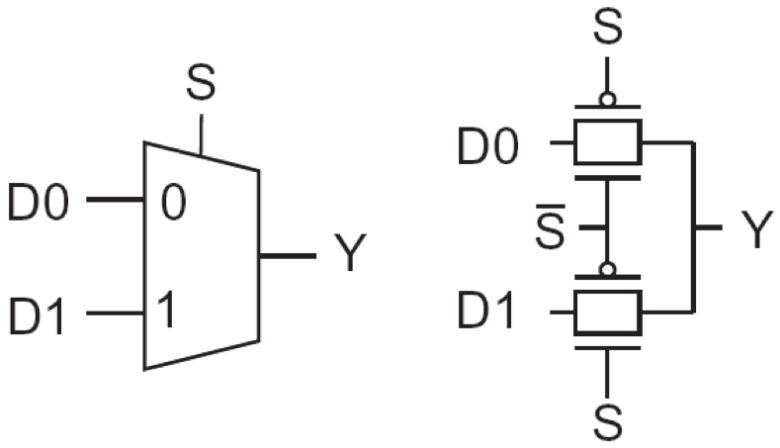


- Separate design into components w/ well-defined interfaces
- Reason, design, and test components in isolation
- Interface may or may not encapsulate implementation

Design Principle: Modularity

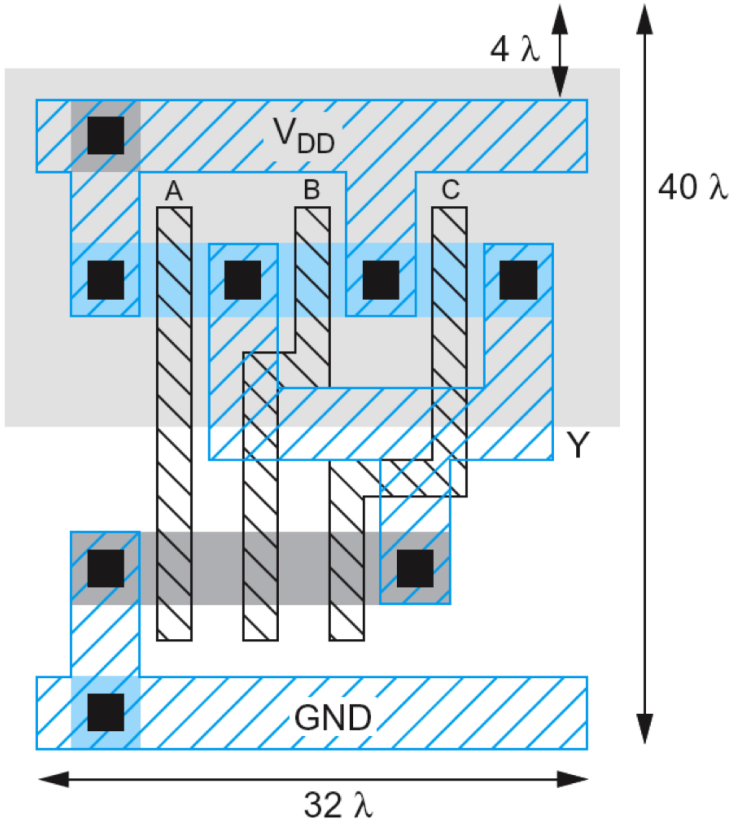
Modularity can also impact electrical and physical characteristics

Electrical Modularity



What happens if we cascade many of these tranmission gate multiplexers?

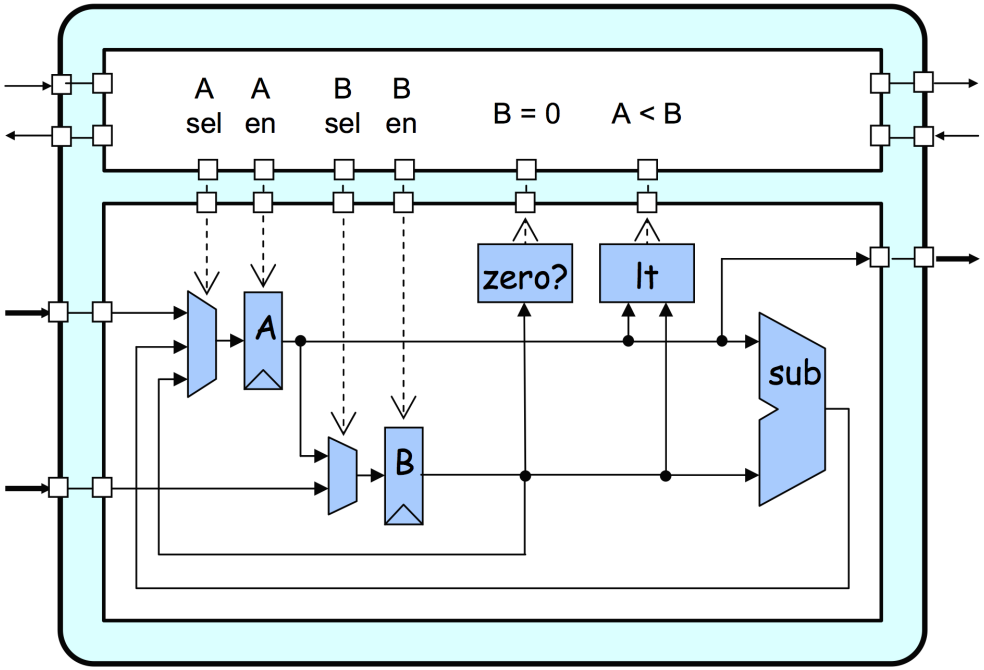
Physical Modularity

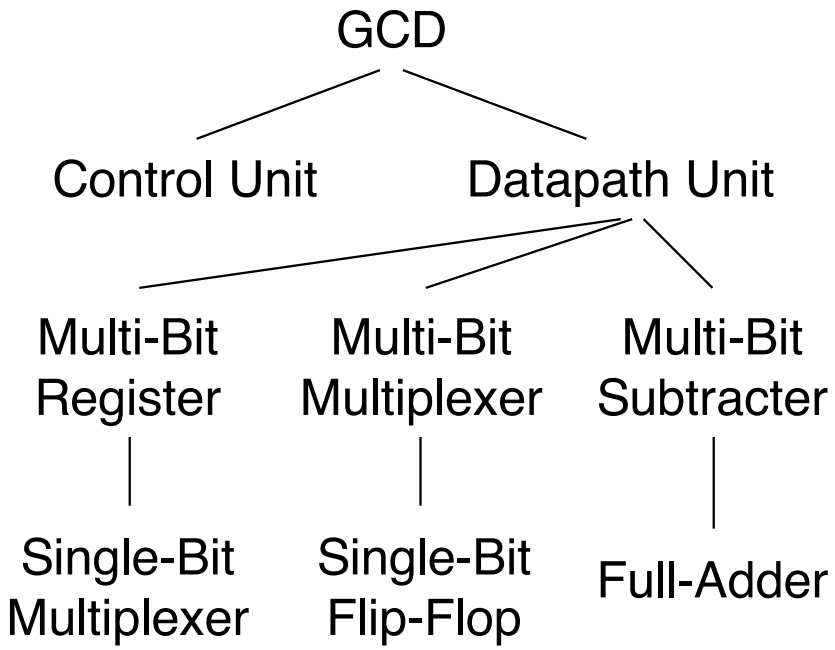


pwr/gnd rails & wells in fixed locations so they connect via abutment

Adapted from [Weste'11]

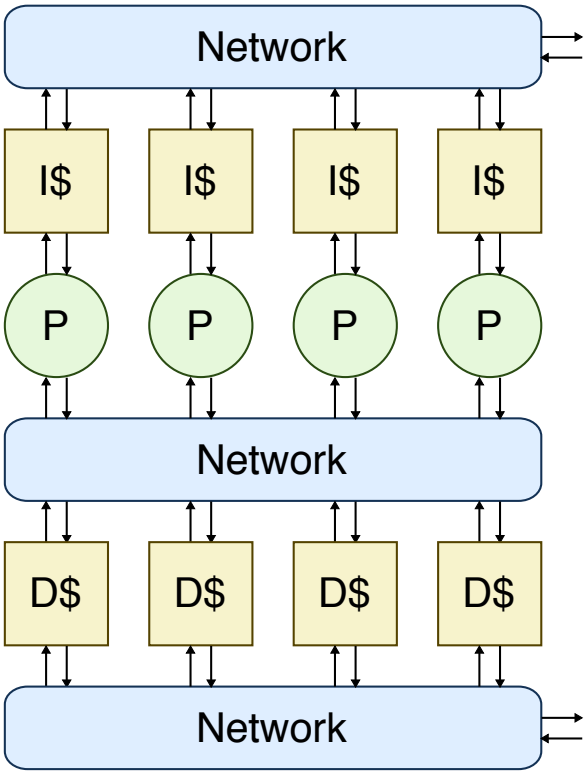
Design Principle: Hierarchy

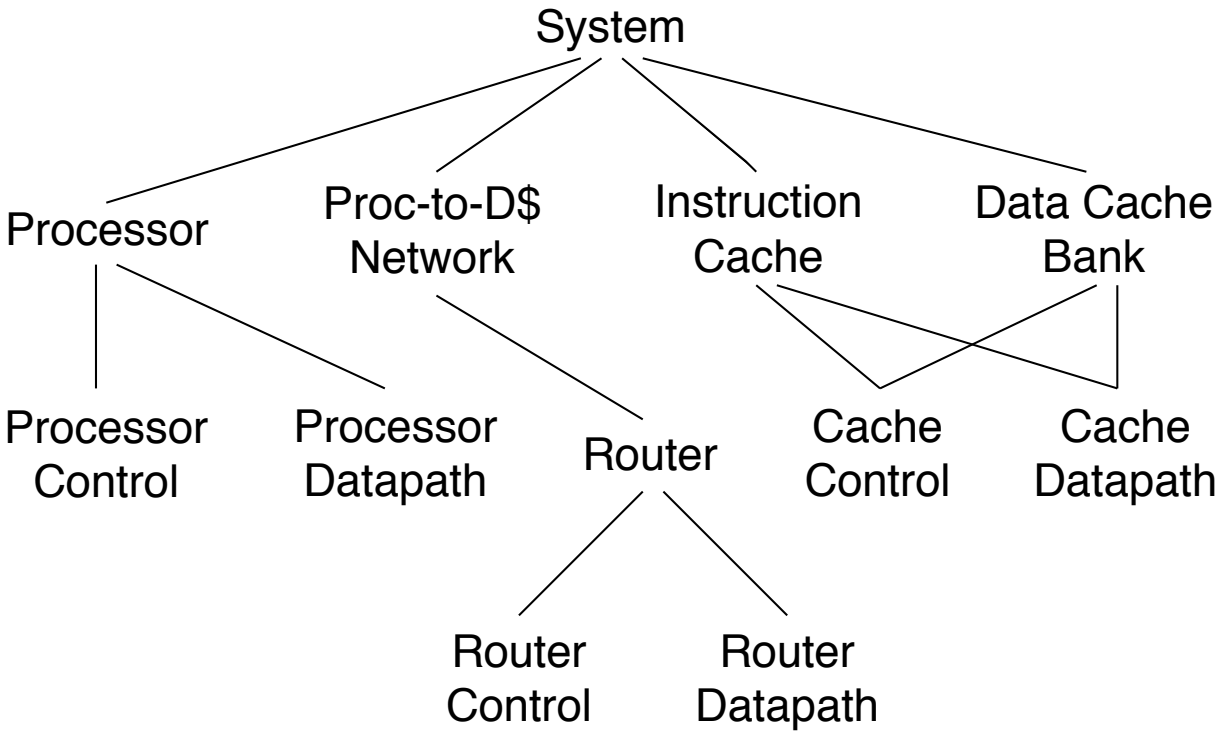




Recursively apply modularity principle until complexity of submodules is manageable

Design Principle: Hierarchy

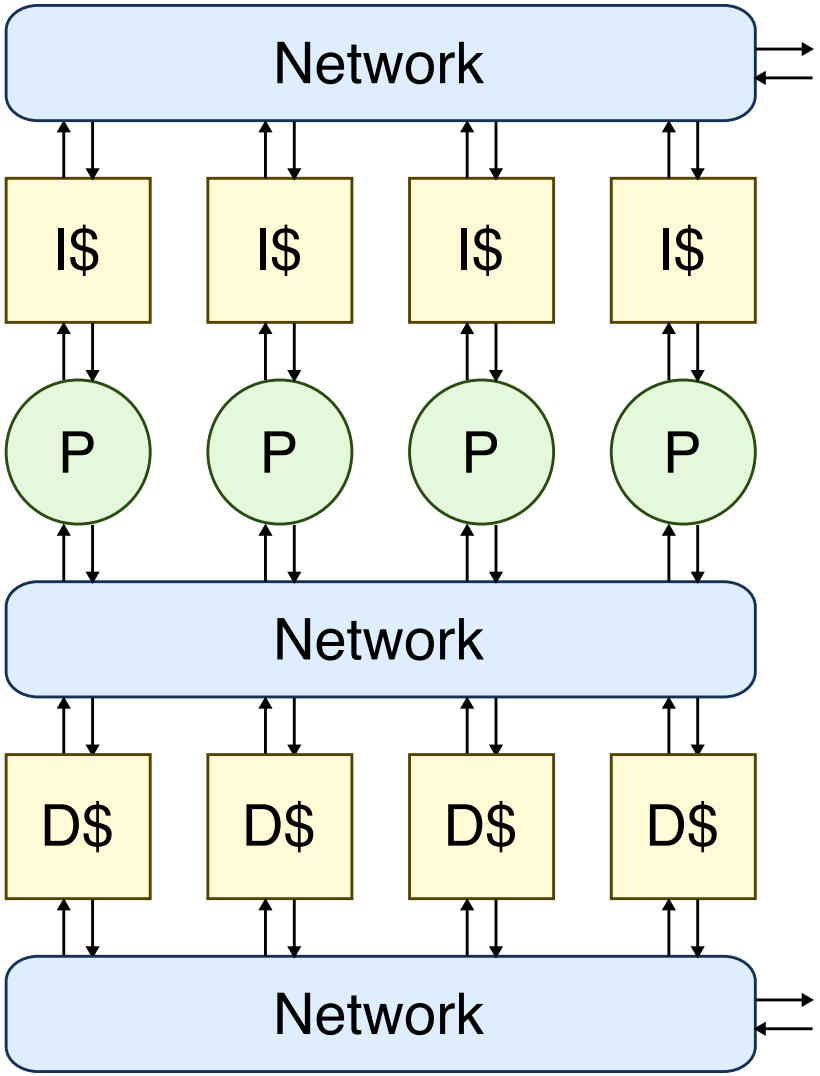




Design Principle: Encapsulation

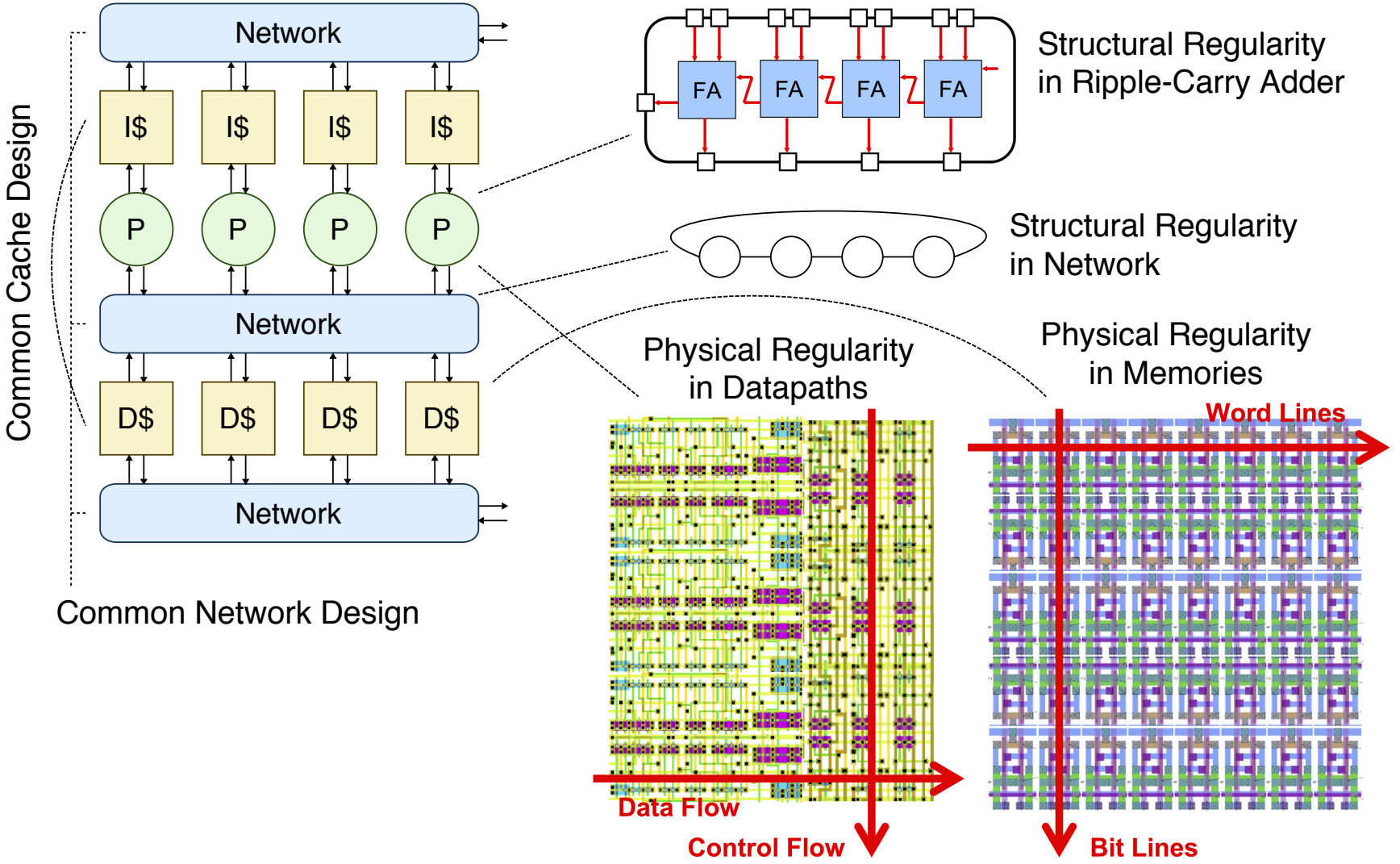
- Modularity requires well-defined interfaces, but these interfaces might still expose significant implementation details (e.g., interface in control/datapath split reveals many details of the implementation)
- Choose interfaces that hide implementation details where possible to enable more robost composition
- Lab 1 multipliers all use a latency-insenstive val/rdy message interface to hide timing details, any one of these can be swaped into a processor and should work without modification
  - Fixed-latency iterative multiplier
  - Variable-latency iterative multplier
  - Pipelined multiplier

Design Principle: Regularity

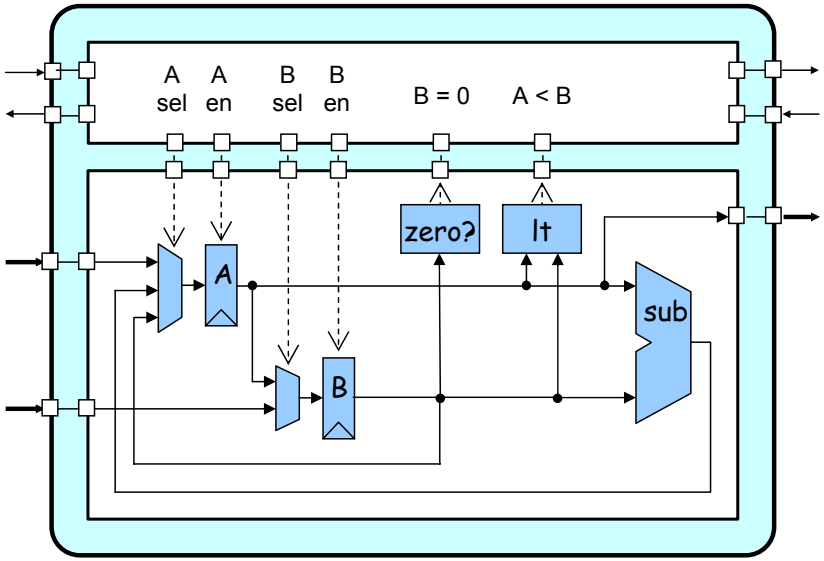


- Modularity, hierarchy, and encapsulation can still lead to many different kinds of modules which can increase design complexity
- Choose a hierarchical decomposition to leverage structure and thus faciliate reuse and reduce complexity
- Both structural and physical regularity can be exploited

Design Principle: Regularity

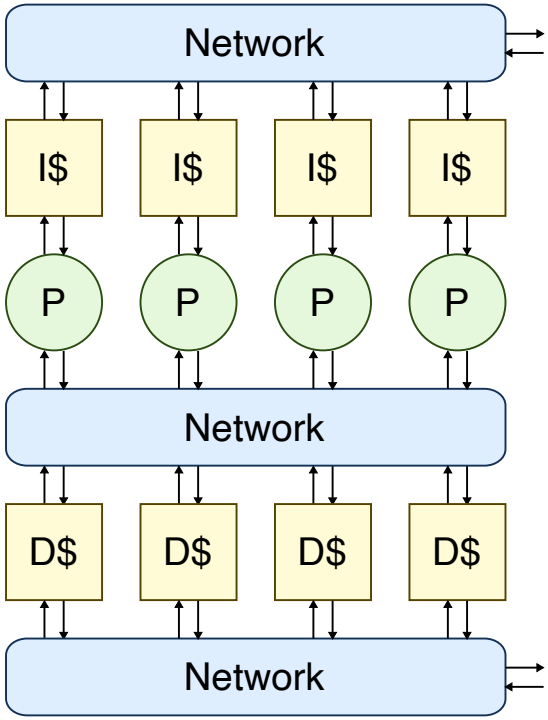


Design Principle: Extensibility



Simple form of polymorphism enables varying bitwidth of operands

Difficult with full-custom design methodology!



Parameterization of network and caches enables reuse; static elaboration could enable varying the number of cores and the types of components

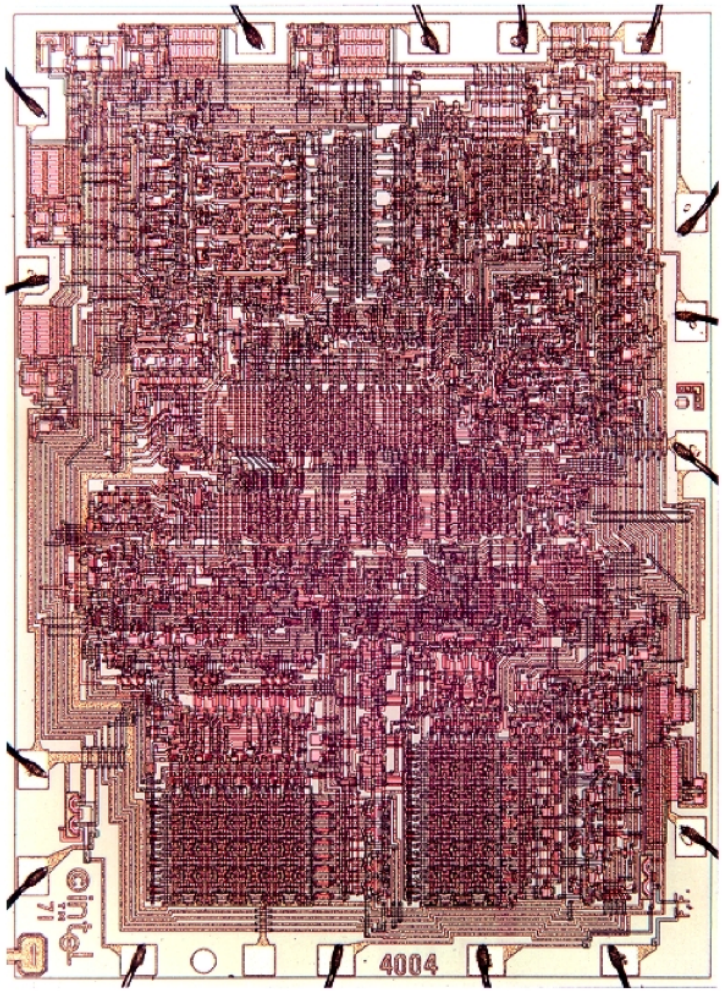
Agenda

Design Domains, Abstractions, and Principles

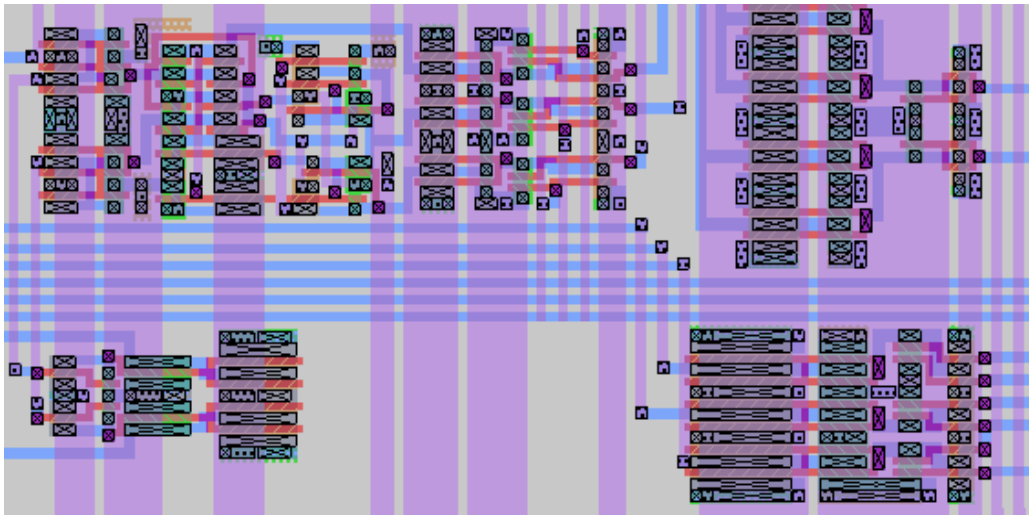
Full-Custom Design

- Cells
- Datapaths
- Memories
- Control

Full Custom Design

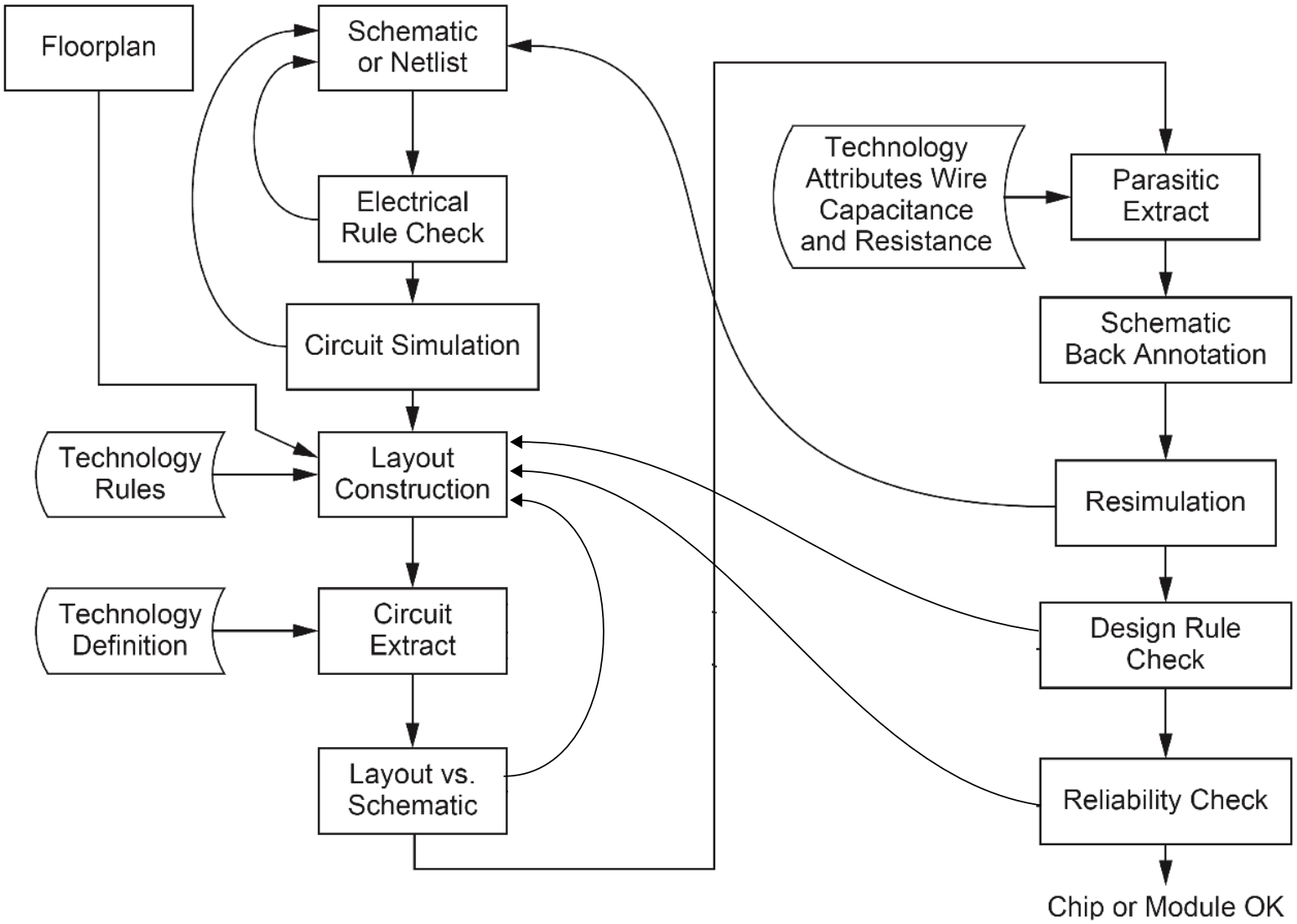


Intel 4004



Key is that all circuits and transistors are optimized for specific context

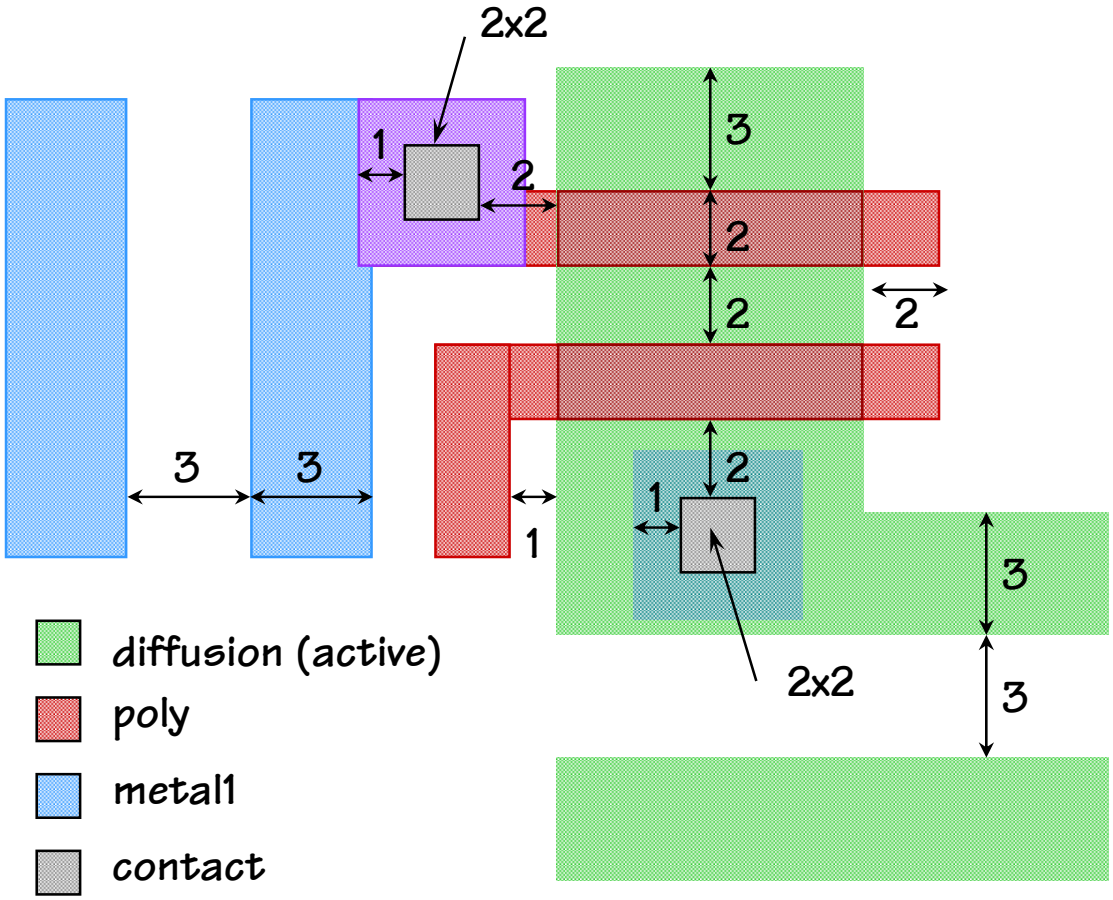
Overview of Full Custom Design Methodology



Adapted from [Weste'11]

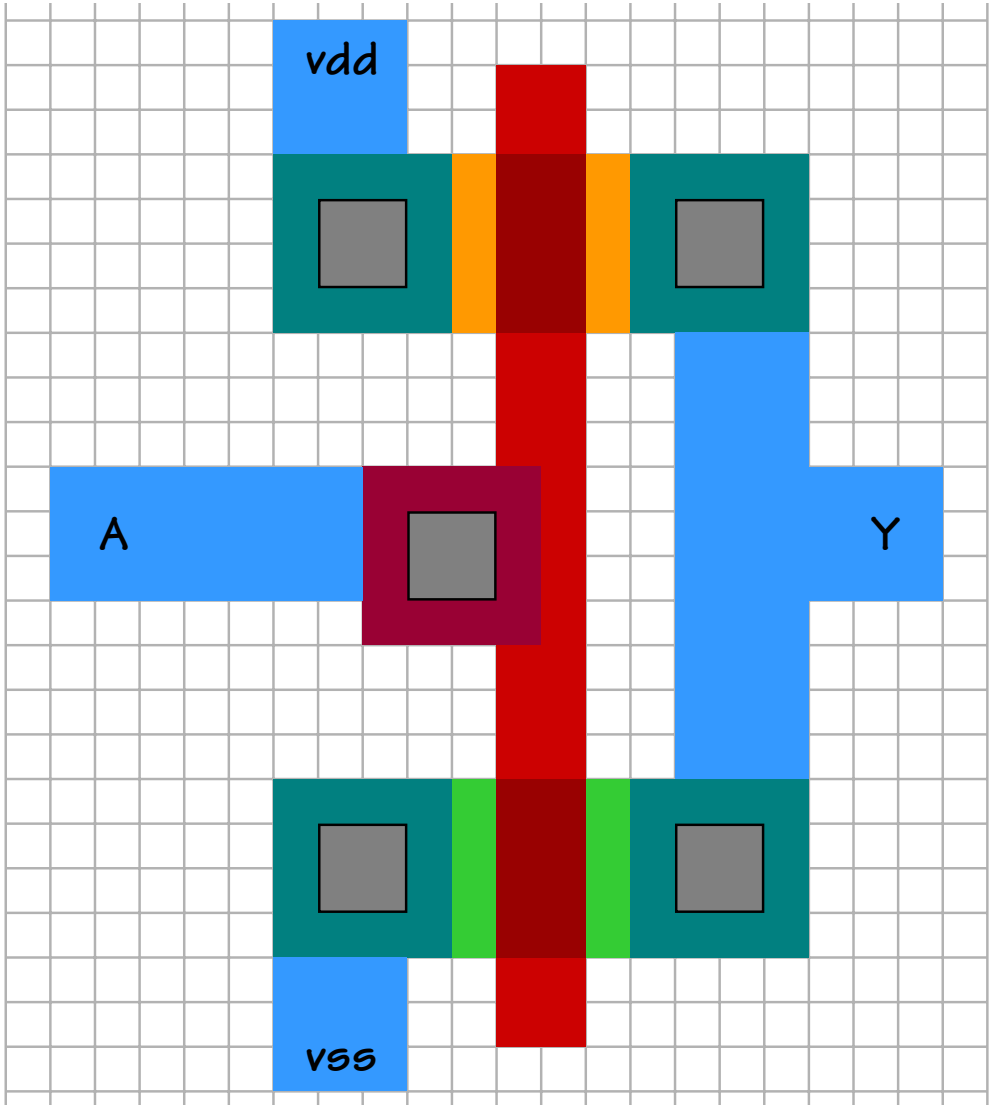
Custom Cells: Lambda-Based Design Rules

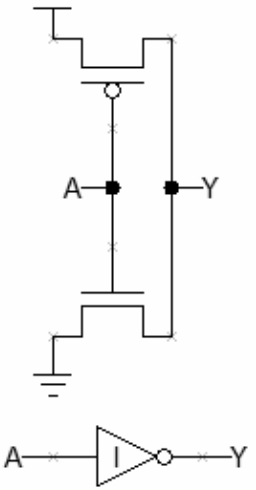
One lambda = one half of the "minimum" mask dimension, typically the length of a transistor channel. Usually all edges must be "on grid", e.g., in the MOSIS scalable rules, all edges must be on a lambda grid.



Adapted from [Terman'02]

Custom Cells: Sample “Lambda” Layout



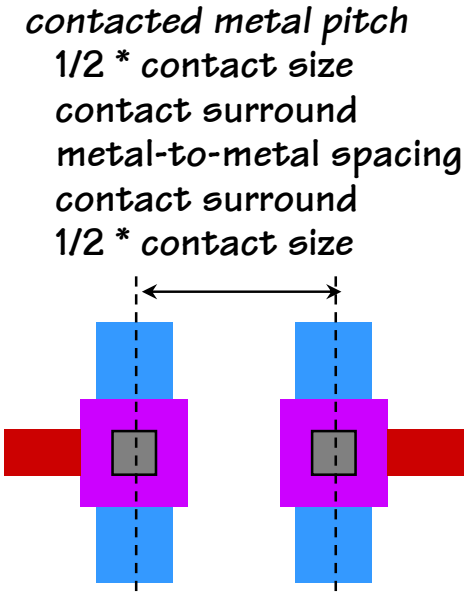


Adapted from [Terman’02]

Custom Cells: Lambda vs. Micron Rules

Lambda-based design rules are based on the assumption that one can scale a design to the appropriate size before manufacture. The assumuption is that all manufacturing dimensions scale equally, an assumption that "works" only over some modest span of time. For example: if a design is completed with a poly width of 2λ and a metal width of 3λ then minimum width metal wires will always be 50% wider than minimum width poly wires.

Consider the following data from *Weste*, Table 3.2:

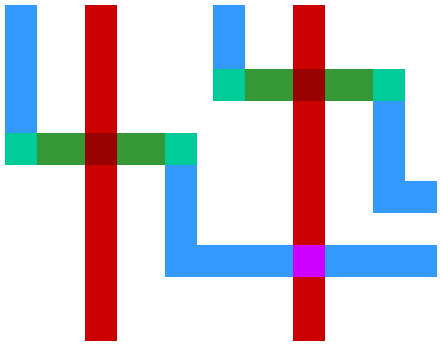


lambda rule	lambda = 0.5u	micron rule
1λ	0.5µ	0.375µ
1λ	0.5µ	0.5µ
3λ	1.5µ	1.0µ
1λ	0.5µ	0.5µ
1λ	0.5µ	0.375µ
7λ	3.5µ	2.75µ

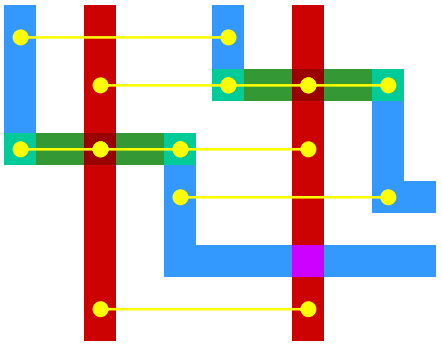
Scaled design is legal but much larger than it needs to be!

Adapted from [Terman'02]

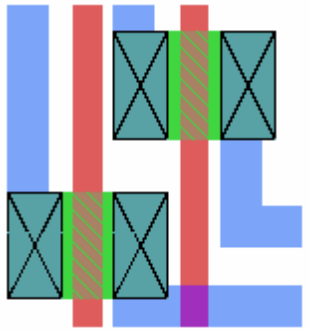
Custom Cells: Sticks and Compaction



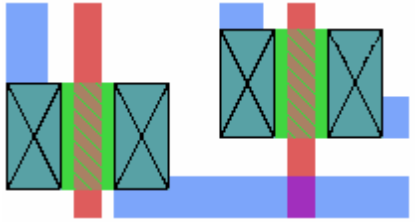
Stick diagram



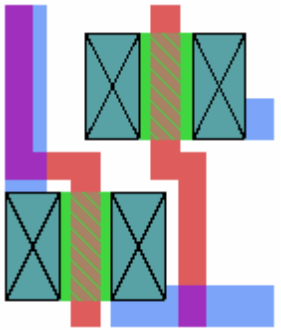
Horizontal constraints for compaction in X



Compact X then Y



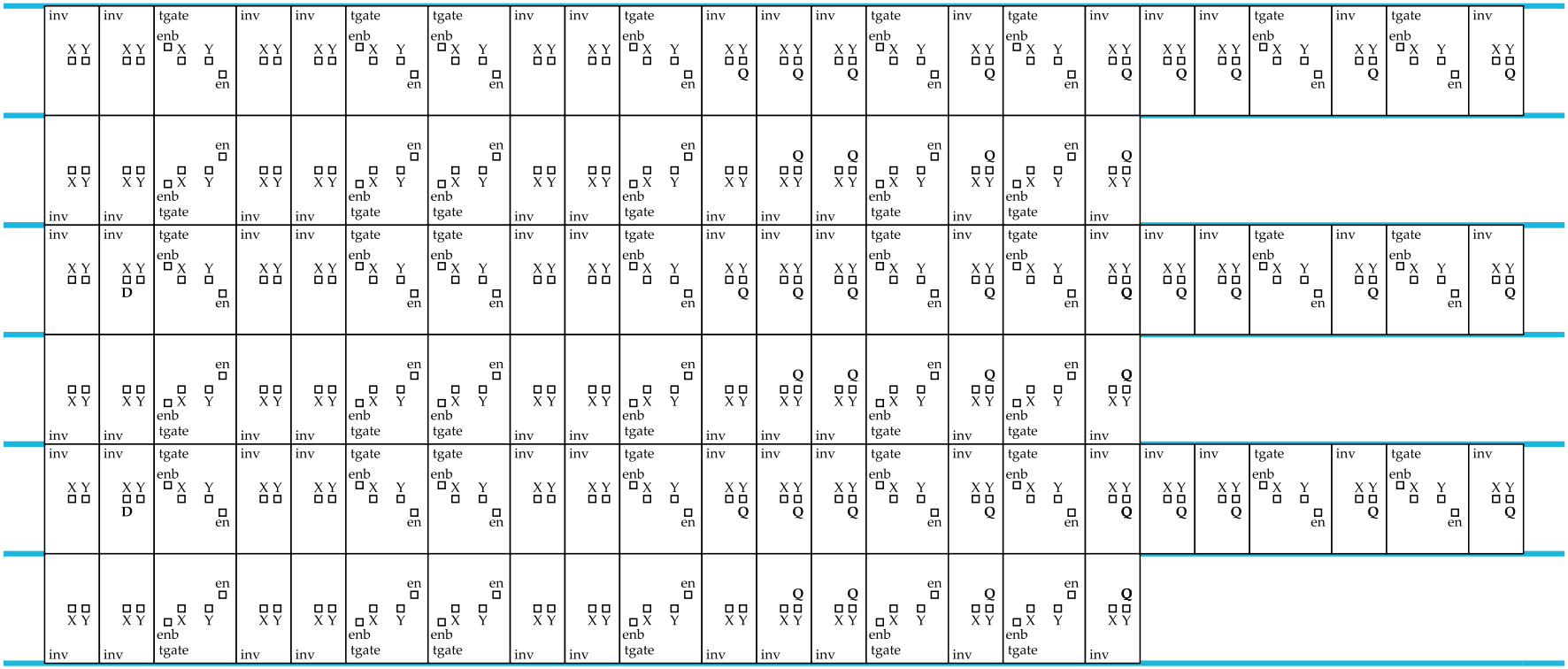
Compact Y then X



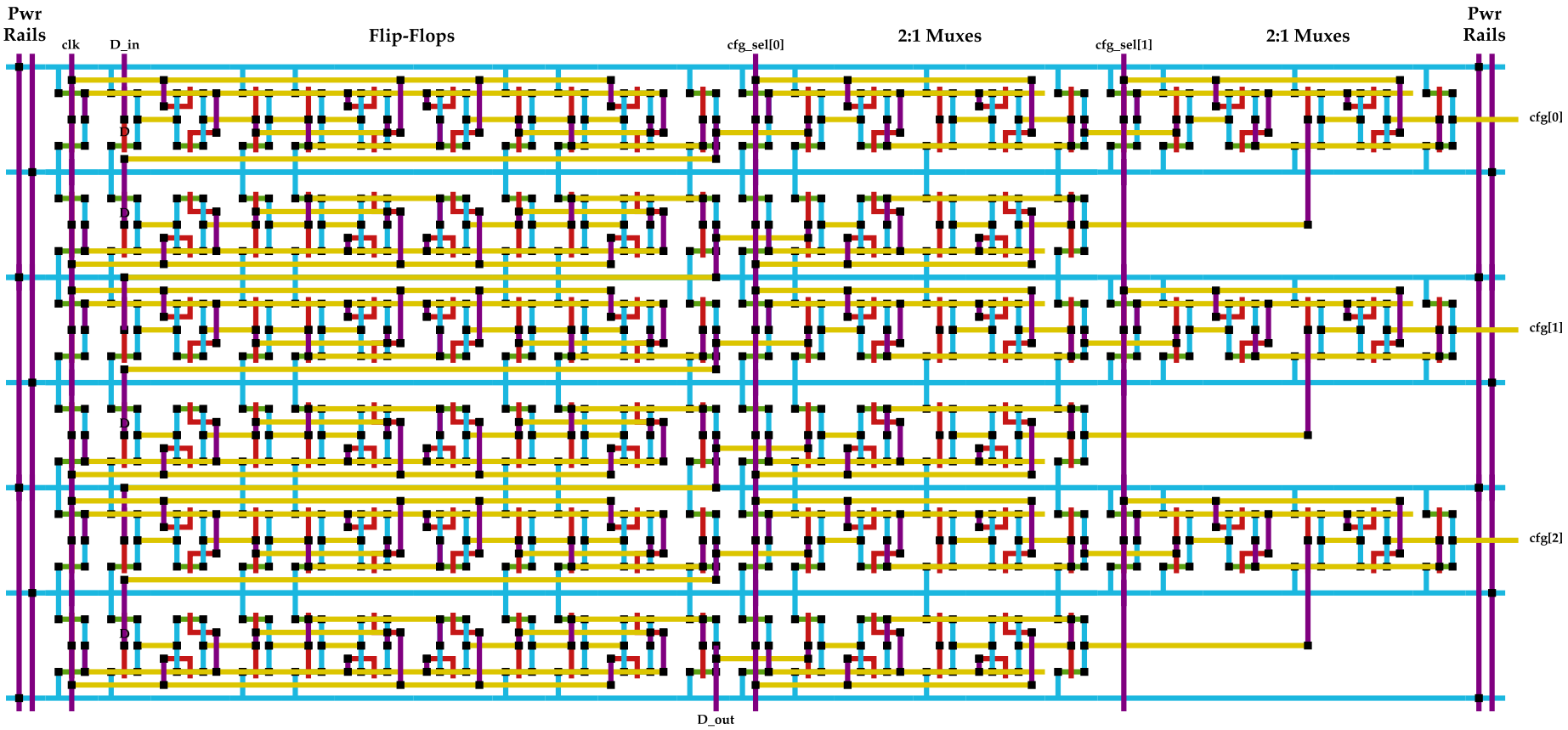
Compact X with jog insertion, then Y

Adapted from [Terman'02]

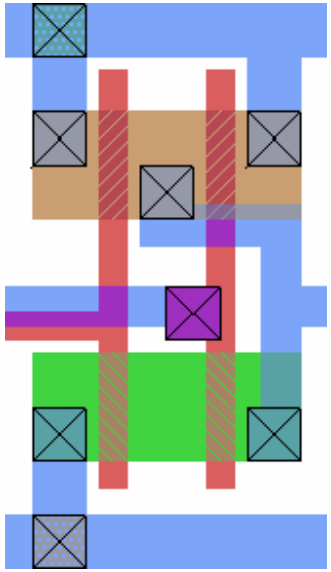
Custom Cells: Example Stick Diagram



Custom Cells: Example Stick Diagram

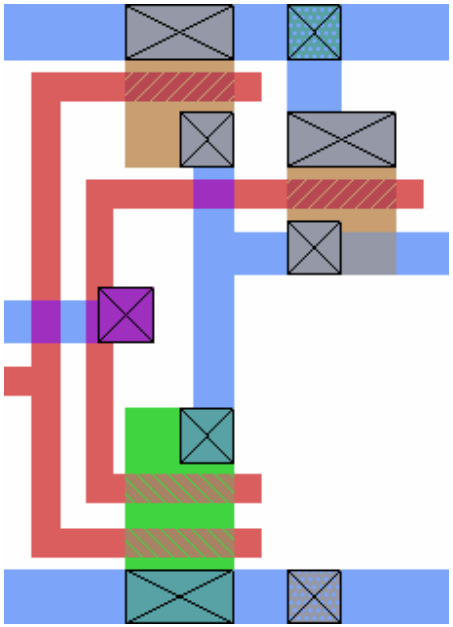


Custom Cells: Cell "Styles"



Vertical Gates

Good for circuits where fets sizes are similar and each gate has limited fanout. Best choice for multiple input static gates and for datapaths.



Horizontal Gates

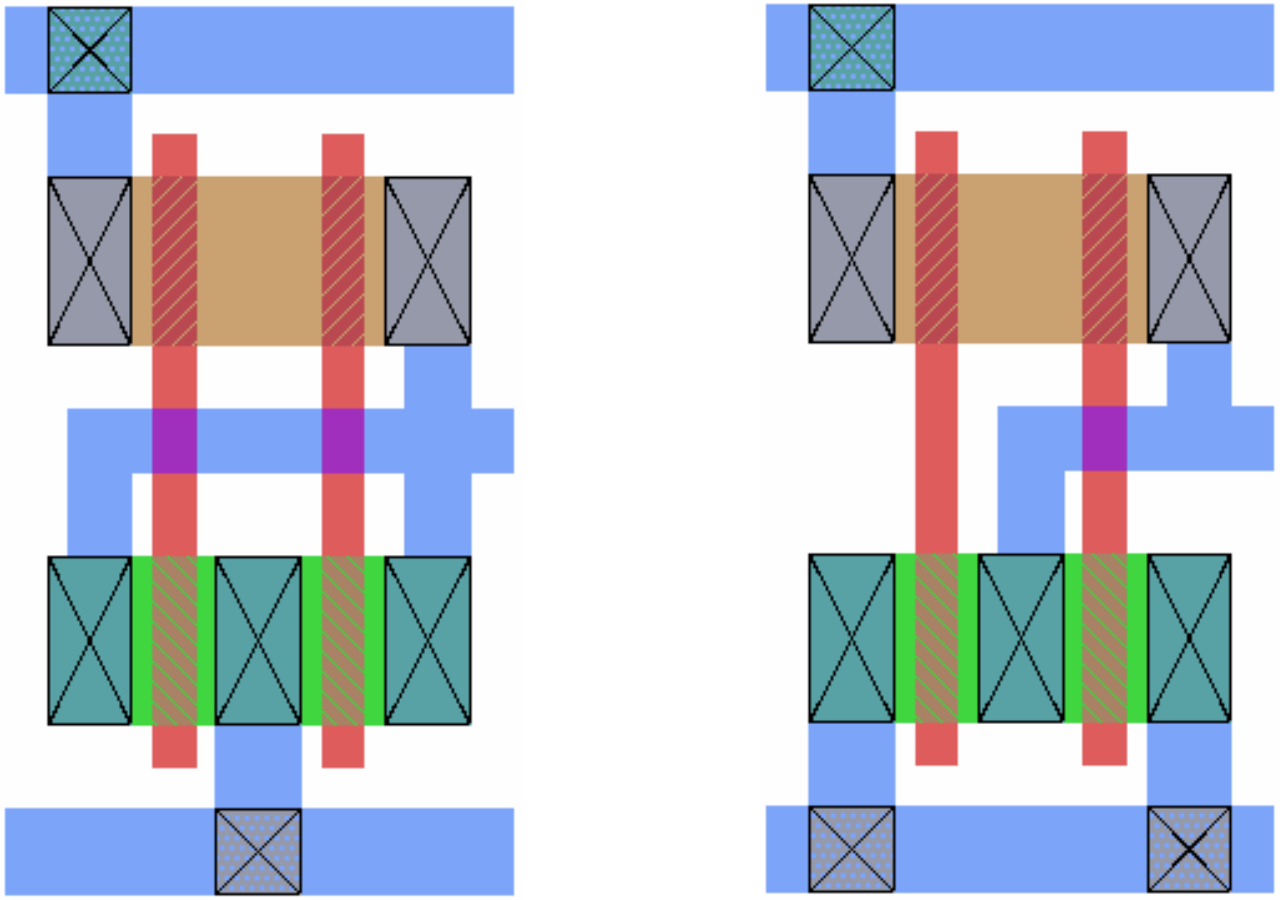
Good for circuits where long and short fets are needed or where nodes must control many fets. Often used in multiple-output complex gates (e.g, sum/carry circuits).

What about routing signals between gates? Note that both layouts block metal/poly routing inside the cell. Choices: metal2 routing over the cell or routing above/below the cell.

- avoid long (> 50 squares) poly runs
- don't "capture" white space in a cell
- don't obsess over the layout, instead make a second pass, optimizing where it counts

Adapted from [Terman'02]

Custom Cells: Optimizing Connections

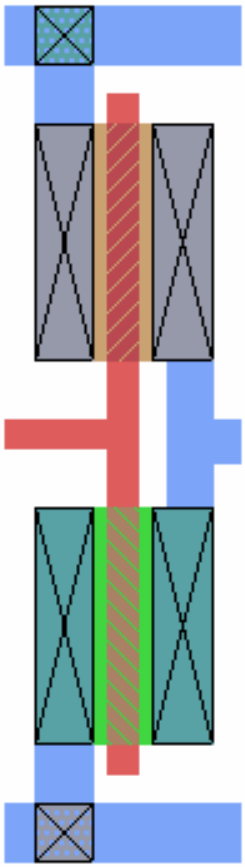


Which does this gate do?

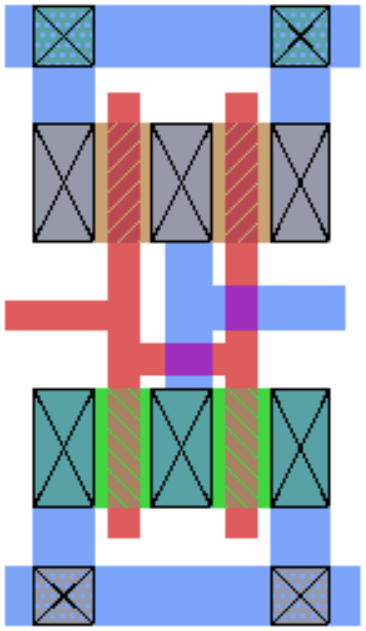
Which is better considering node capacitances?

Adapted from [Terman'02]

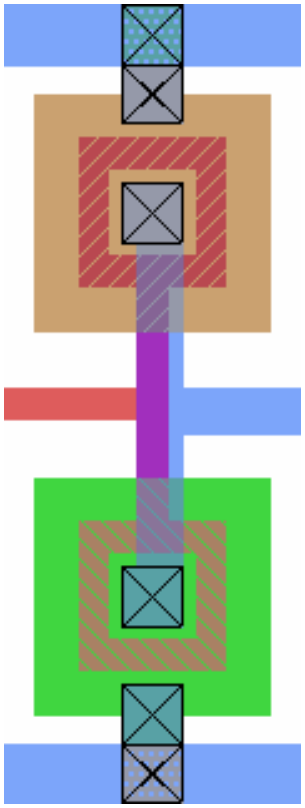
Custom Cells: Optimizing Large Transistors



area = 928



area = 1008

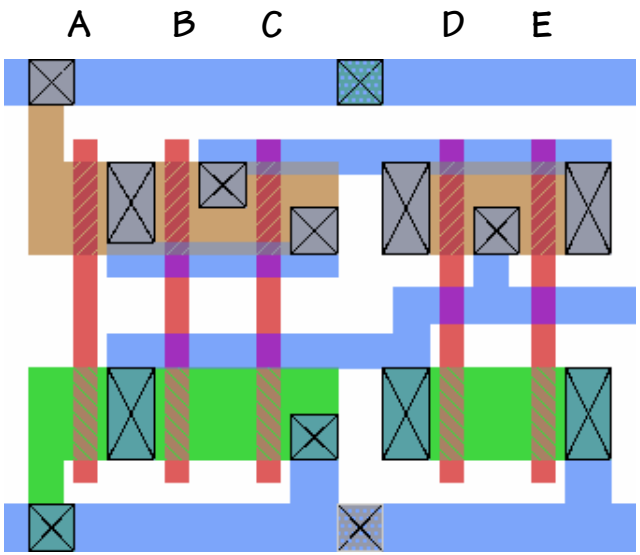


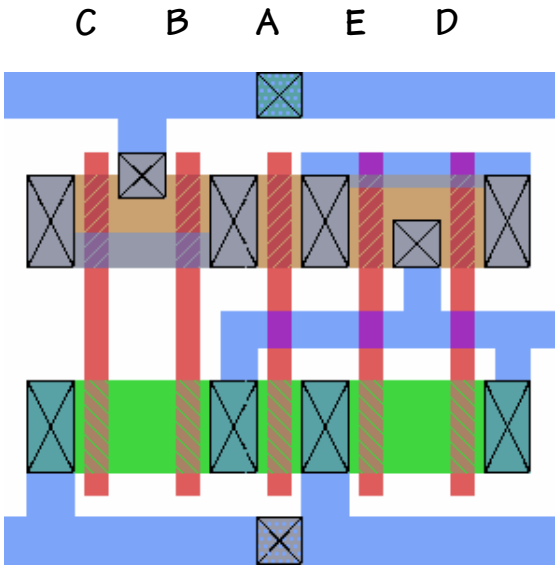
area = 1080

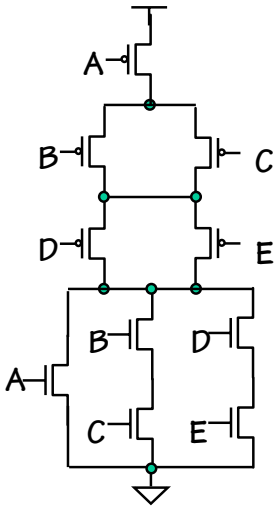
Which is better considering wire resistances?
Which is better considering node capacitances?

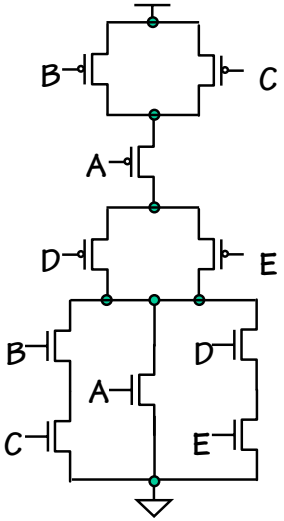
Adapted from [Terman'02]

Custom Cells: Optimizing Diffusion Sharing



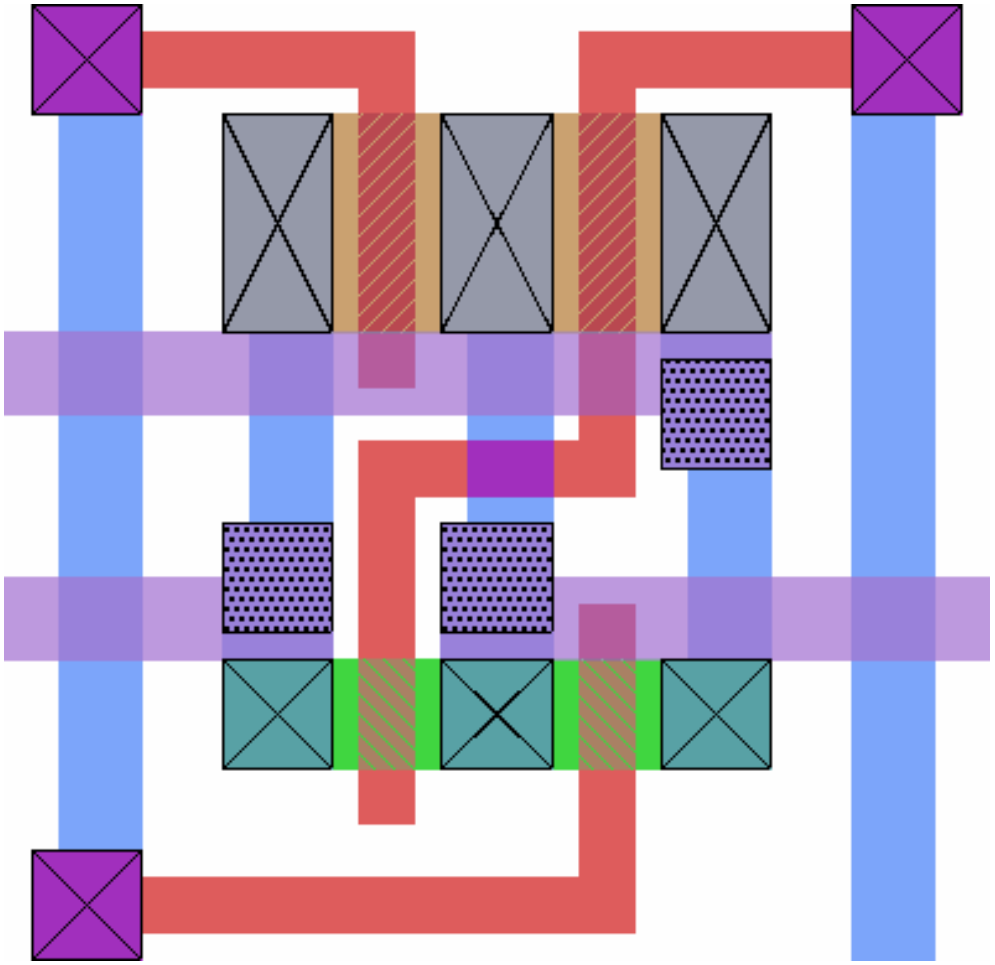






Adapted from [Terman'02]

Custom Cells: Optimizing Across Cells

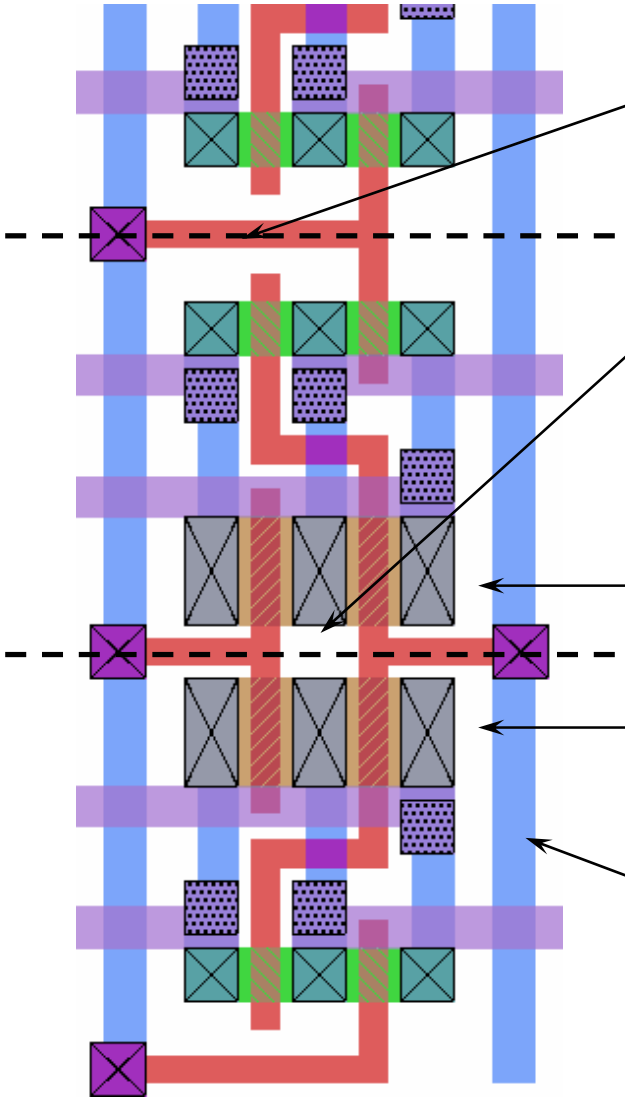


What does this cell do?

What if we want to replicate this cell vertically to process many bits in parallel?

Adapted from [Terman'02]

Custom Cells: Optimizing Across Cells



Place shared geometry symmetrically about shared boundary.

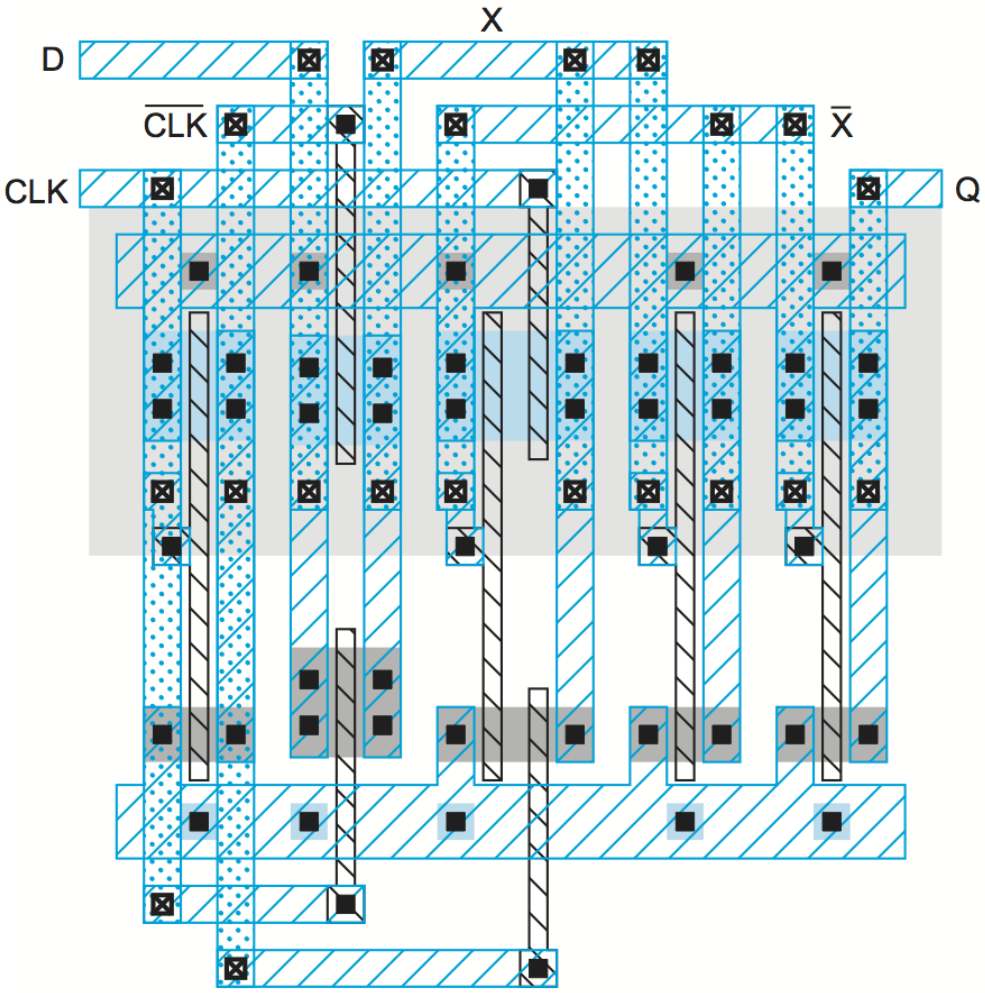
Place items that aren't to be shared 1/2 minspacing rule from shared boundary.

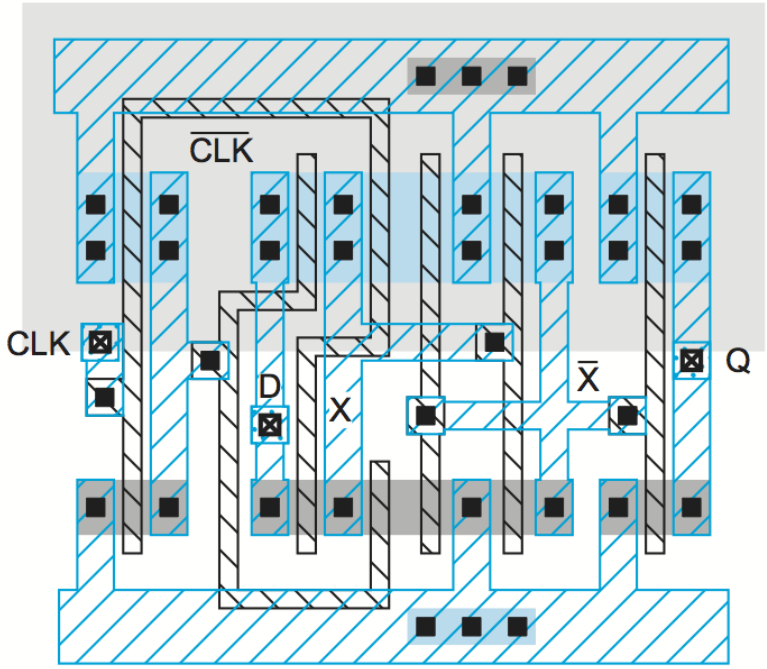
Reflect cell about X axis so that Pfets are next to each other: this avoids large ndiff/pdiff spacing.

Run shared control signals vertically -- they'll wire themselves up automatically?

Adapted from [Terman'02]

Custom Cells: Merging Simple Cells

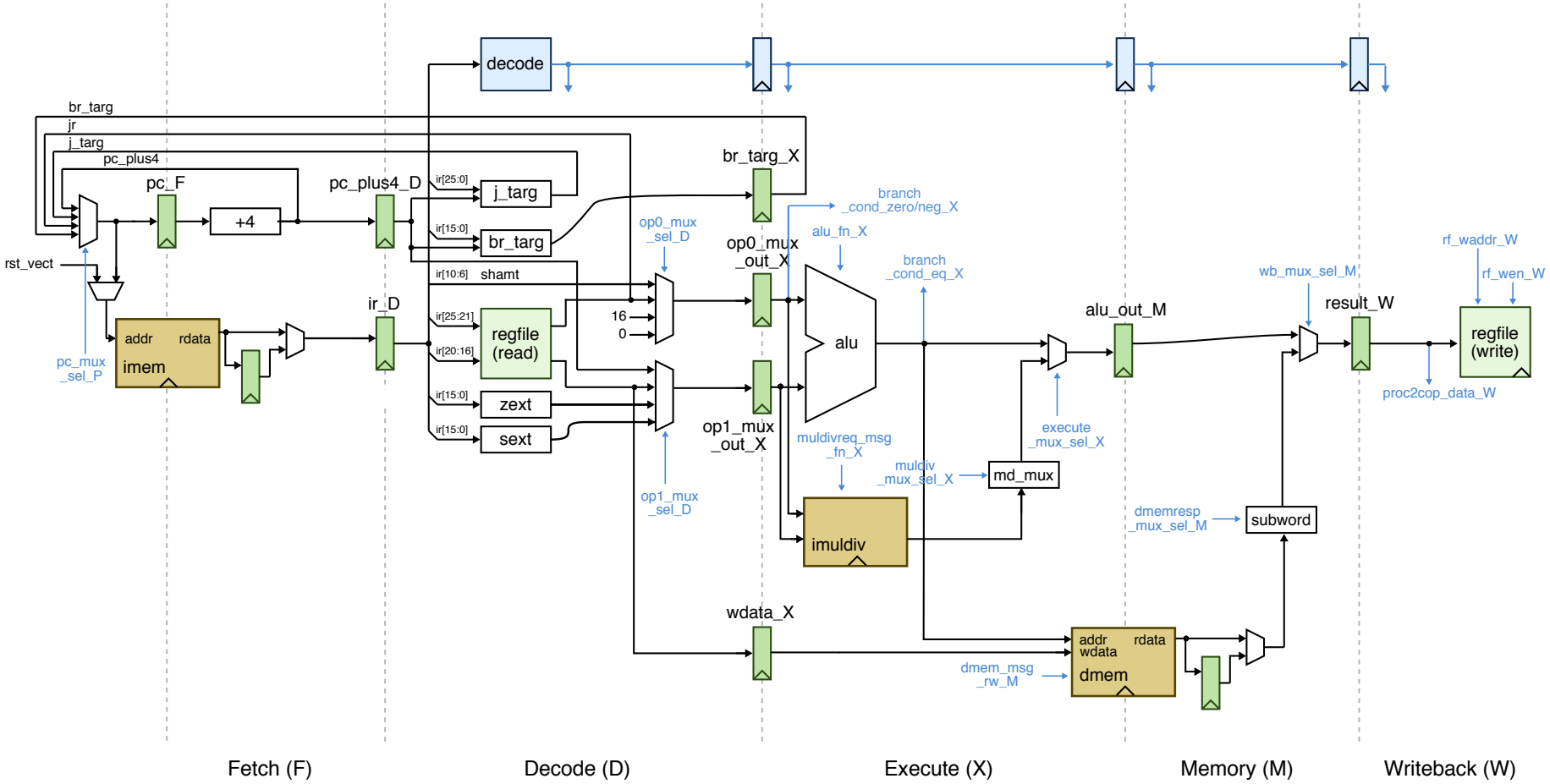




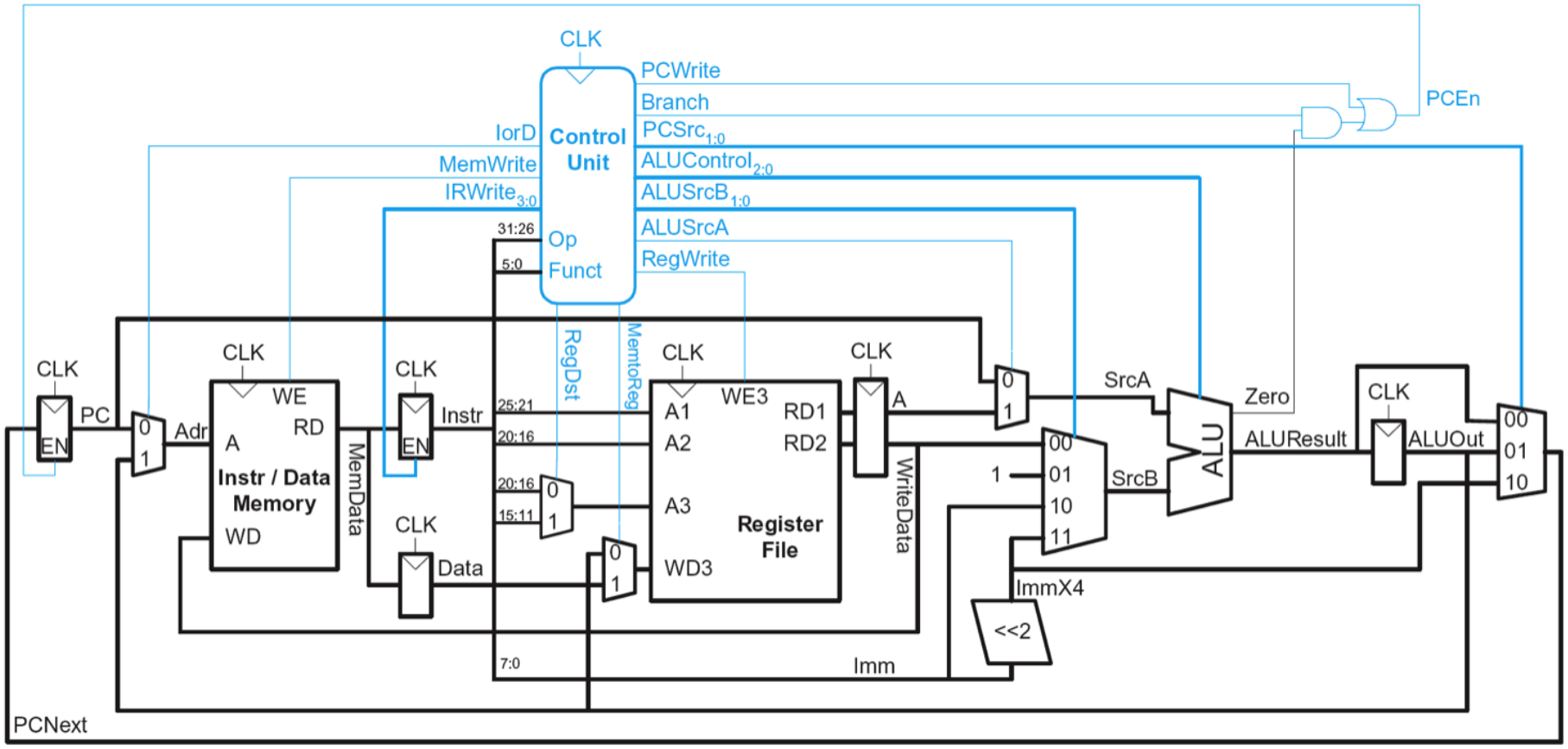
Two latch implementations: Left implementation composes primitive gates, while right implementation uses single tightly integrated gate

Adapted from [Weste'11]

Custom Datapaths, Memories, Control

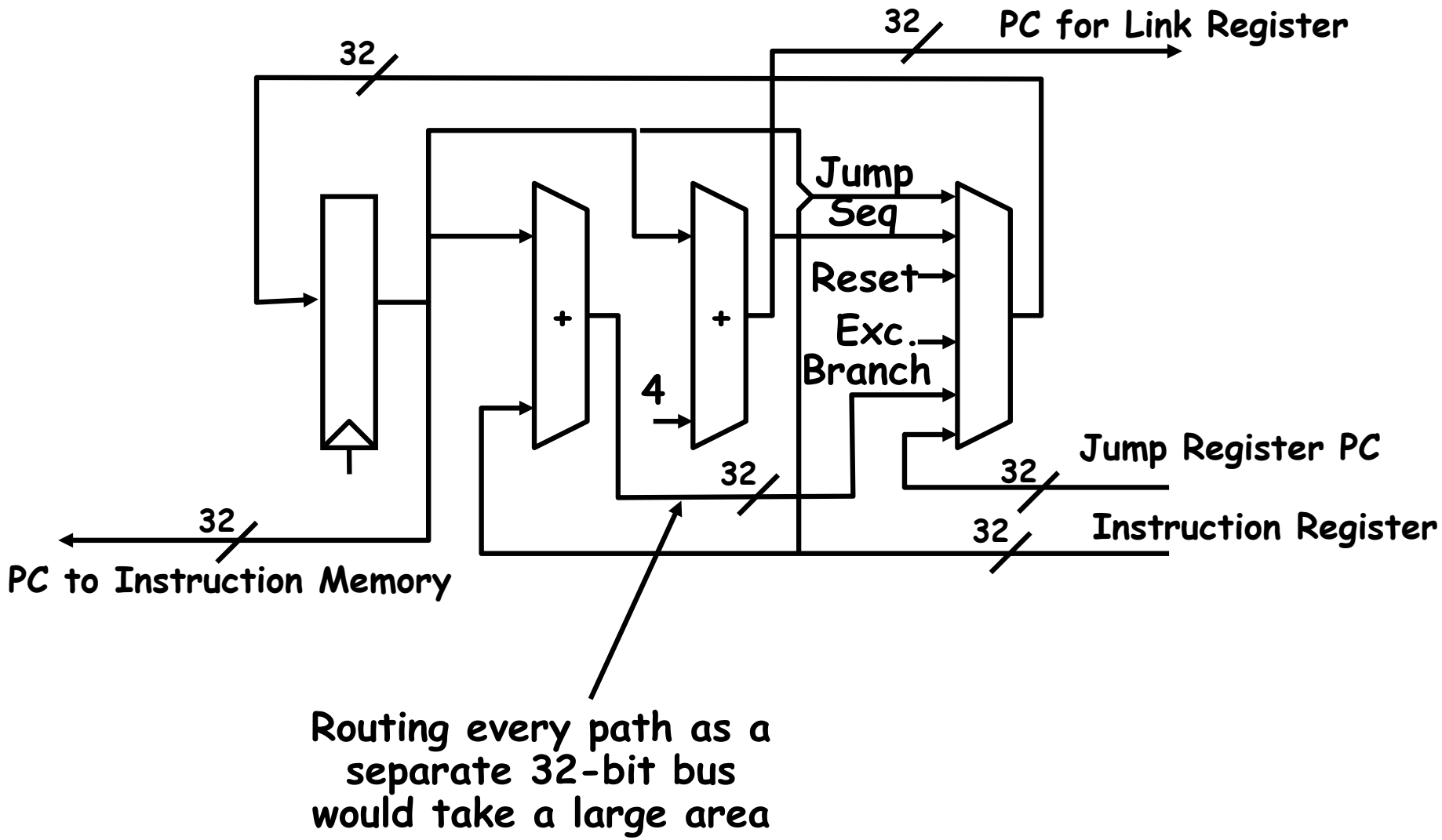


Custom Datapaths, Memories, Control



Adapted from [Weste'11]

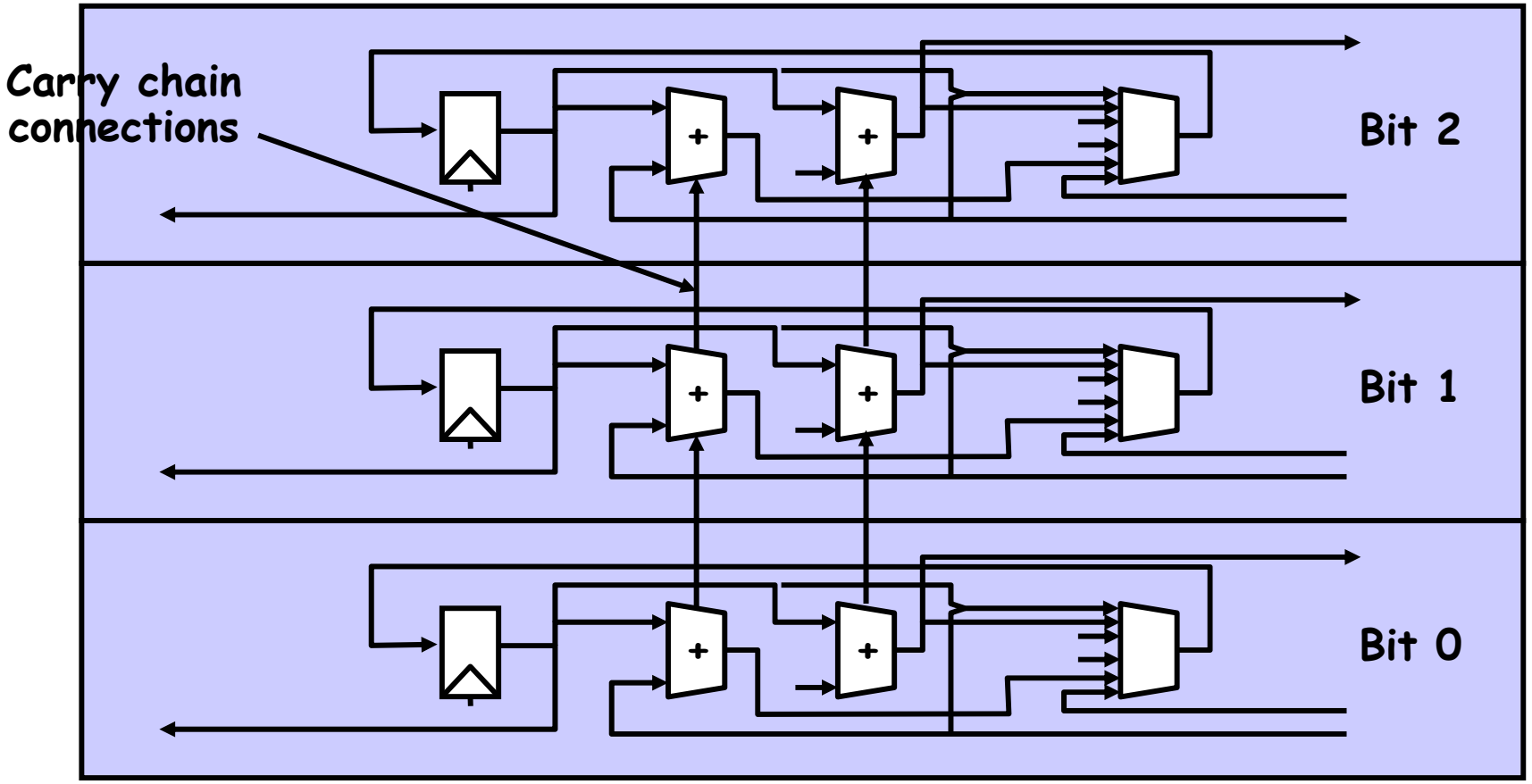
Custom Datapaths: PC Generation Unit



Adapted from [Terman'02]

Custom Datapaths: Bitslices

- Implement datapath as single bit slices, contain one bit from each functional unit
- Route each bus bit position within bitslice

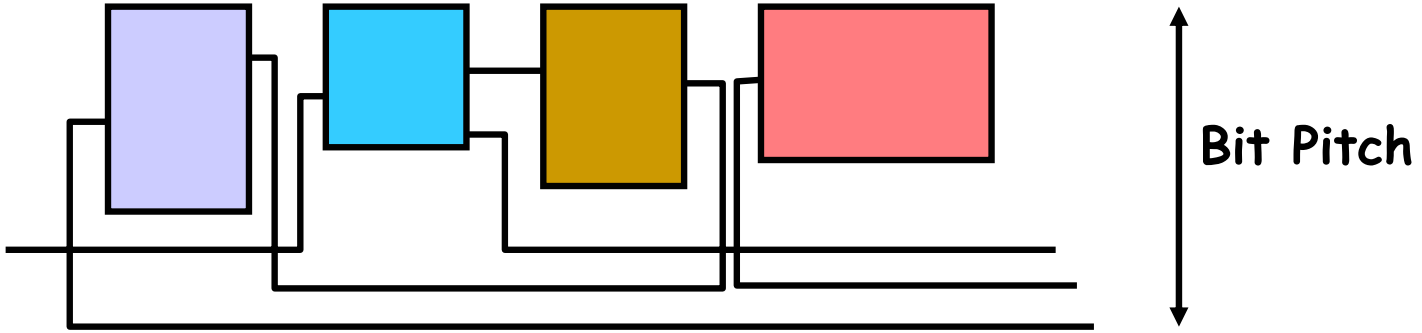


Adapted from [Terman'02]

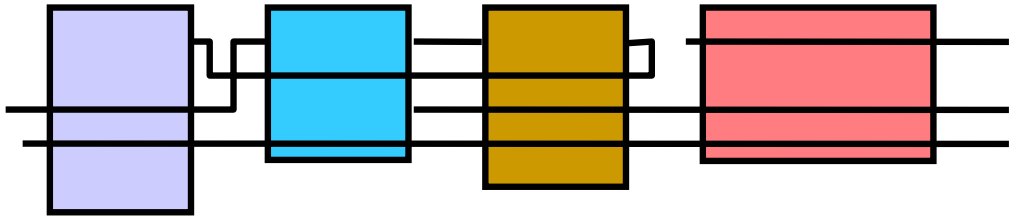
Custom Datapaths: Bit Pitch

- Height of each bit slice depends on:
  - height of tallest cell in entire bitslice
  - maximum number of buses running through any cell in bitslice

Two layers metal, buses run by side of cells

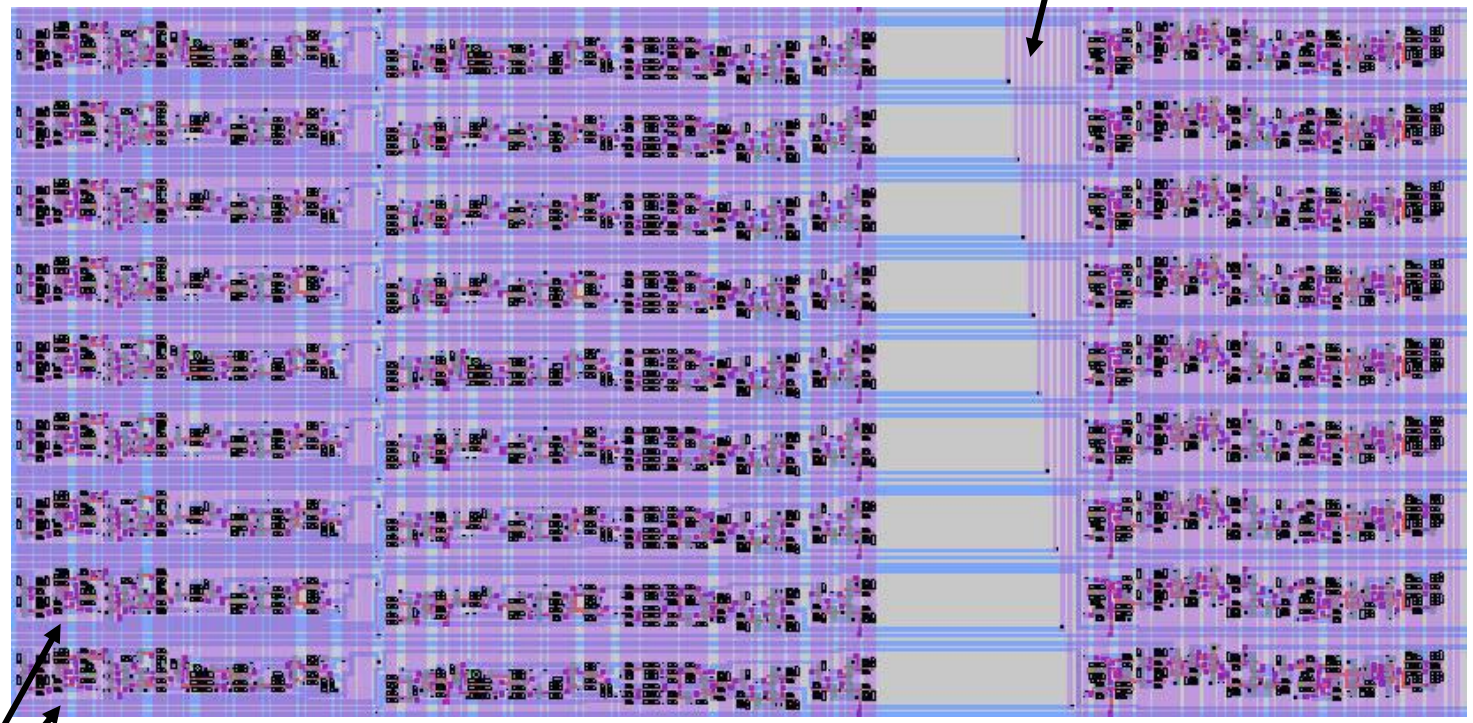


Three+ layers metal, buses run over cells



Adapted from [Terman'02]

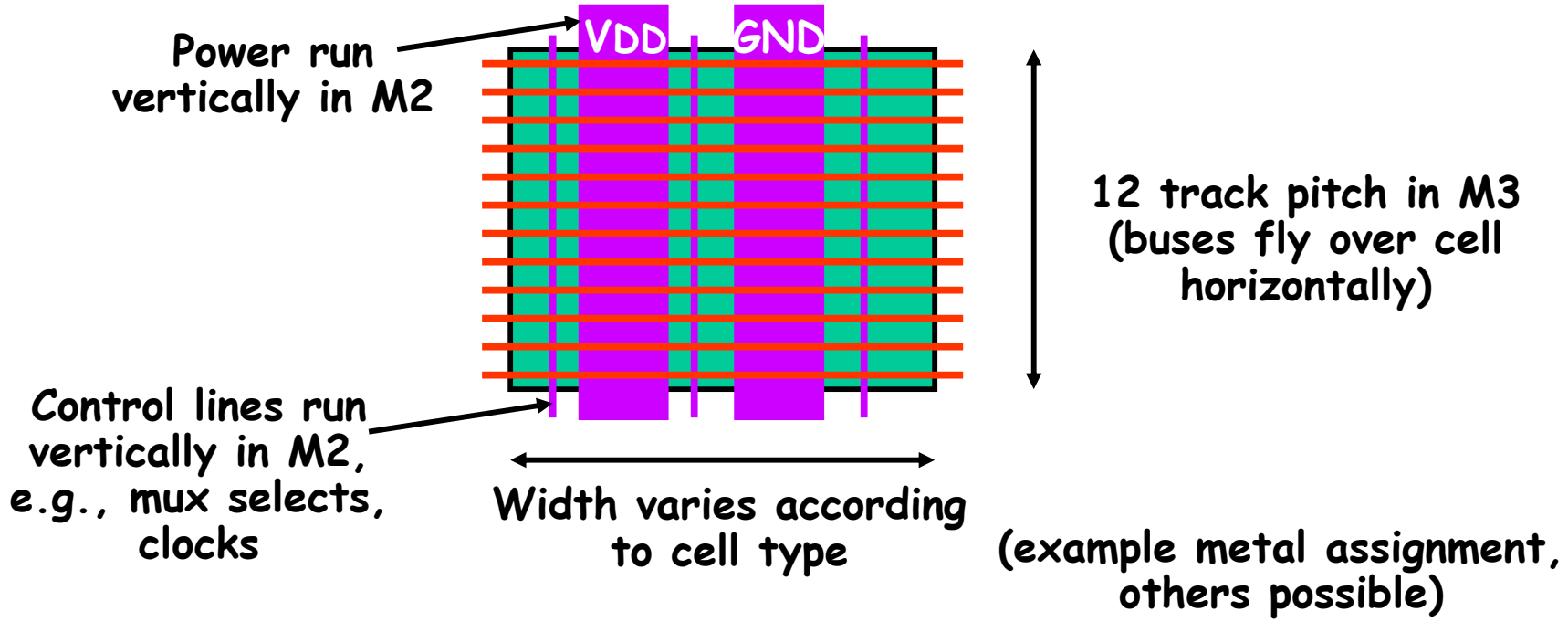
Custom Datapaths: PC Gen Example



Adapted from [Terman'02]

Custom Datapaths: Datapath Library Cells

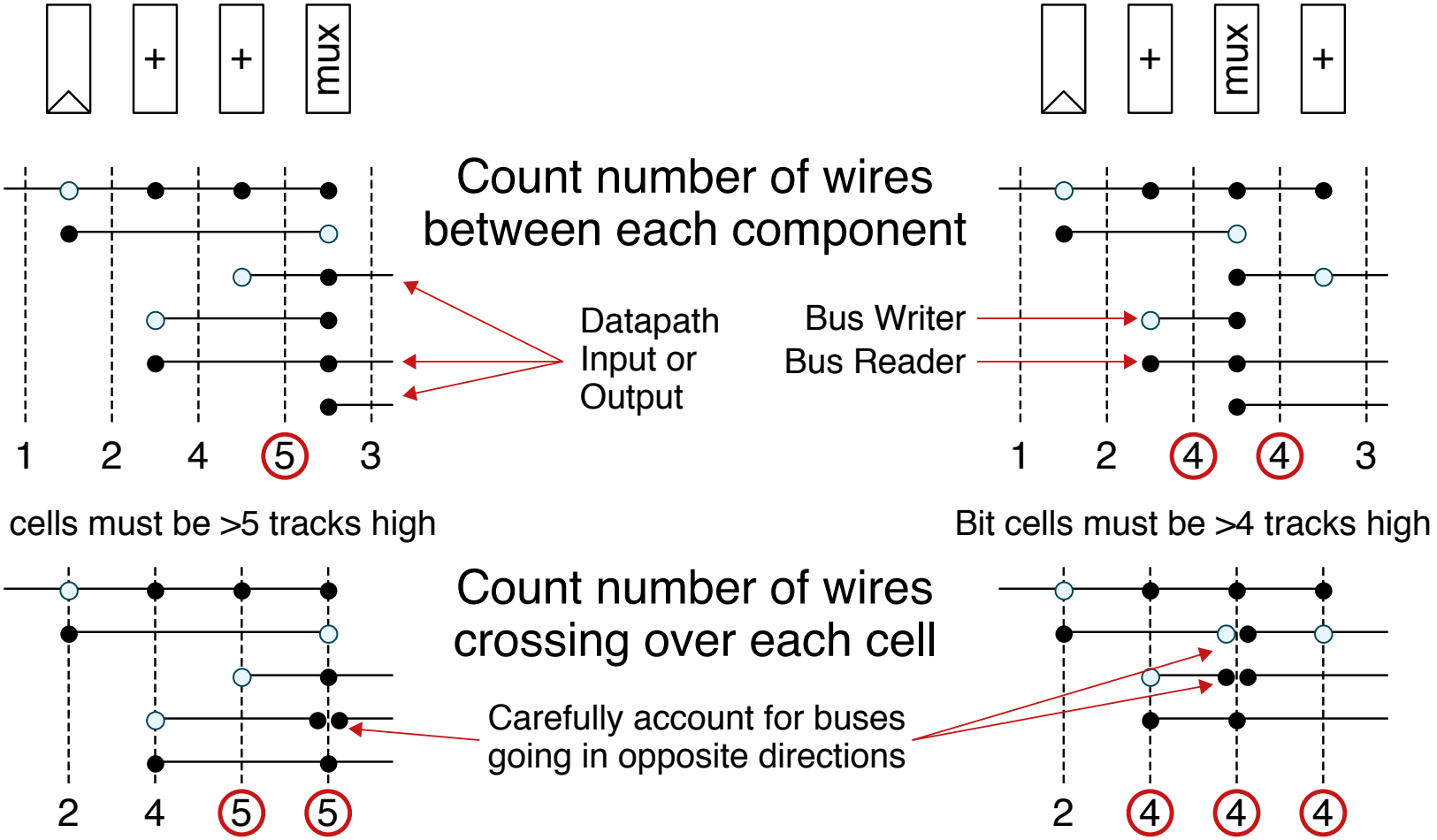
- Have to choose maximum datapath cell height
  - Too high, wastes area in simple cells
  - Too small, squeezes complex cells. Grow superlinearly in length dimension, so also wastes area.
- Compromise, around 8-12 metal tracks works OK



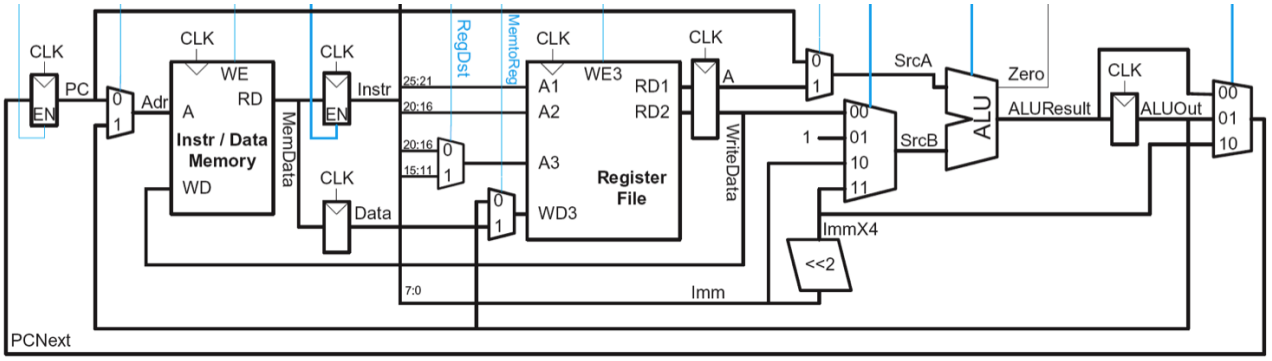
Adapted from [Terman'02]

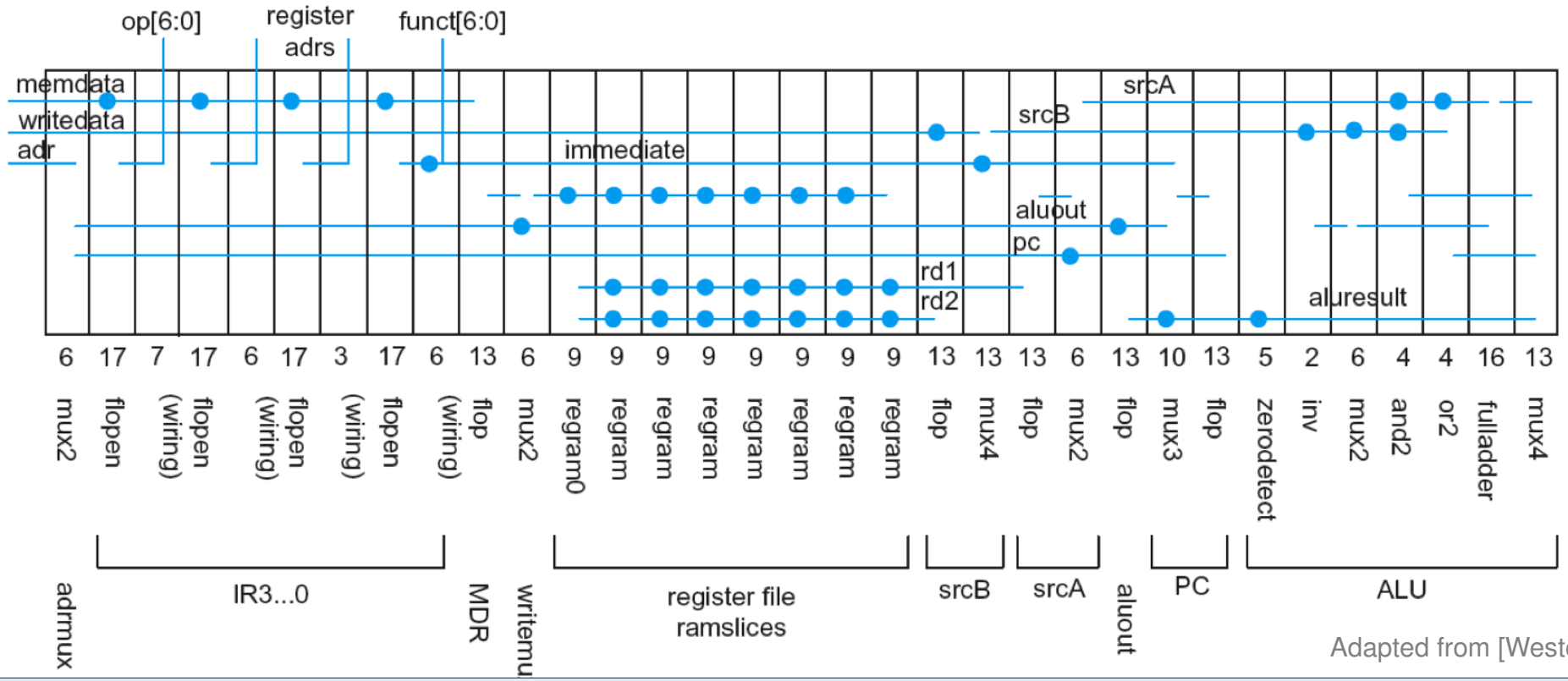
Custom Datapaths: Optimizing Datapath Layout

Reduce congestion by rearranging datapath components to minimize required number of vertical tracks per bitslice



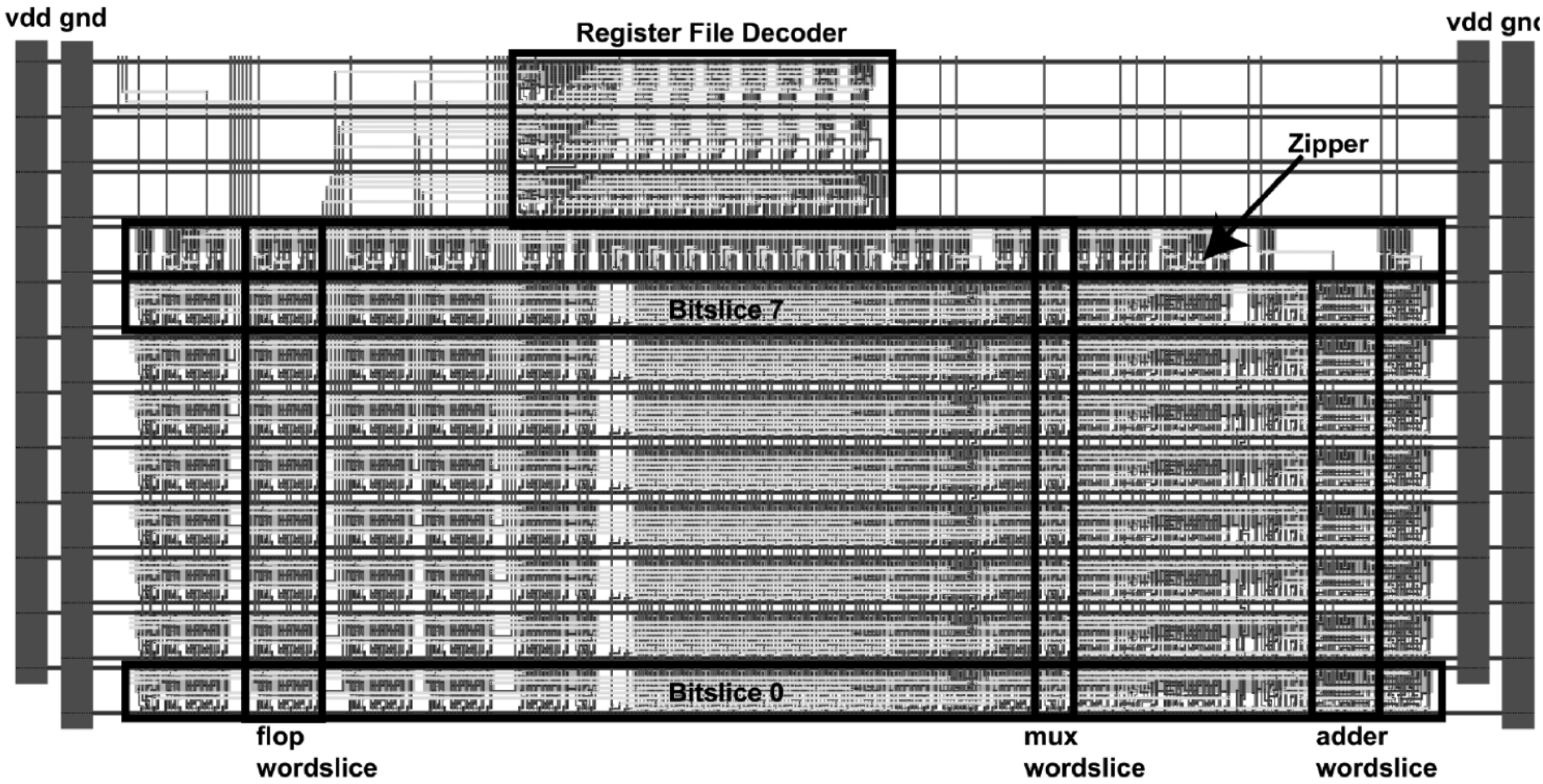
Custom Datapaths: MIPS Datapath Track Allocation





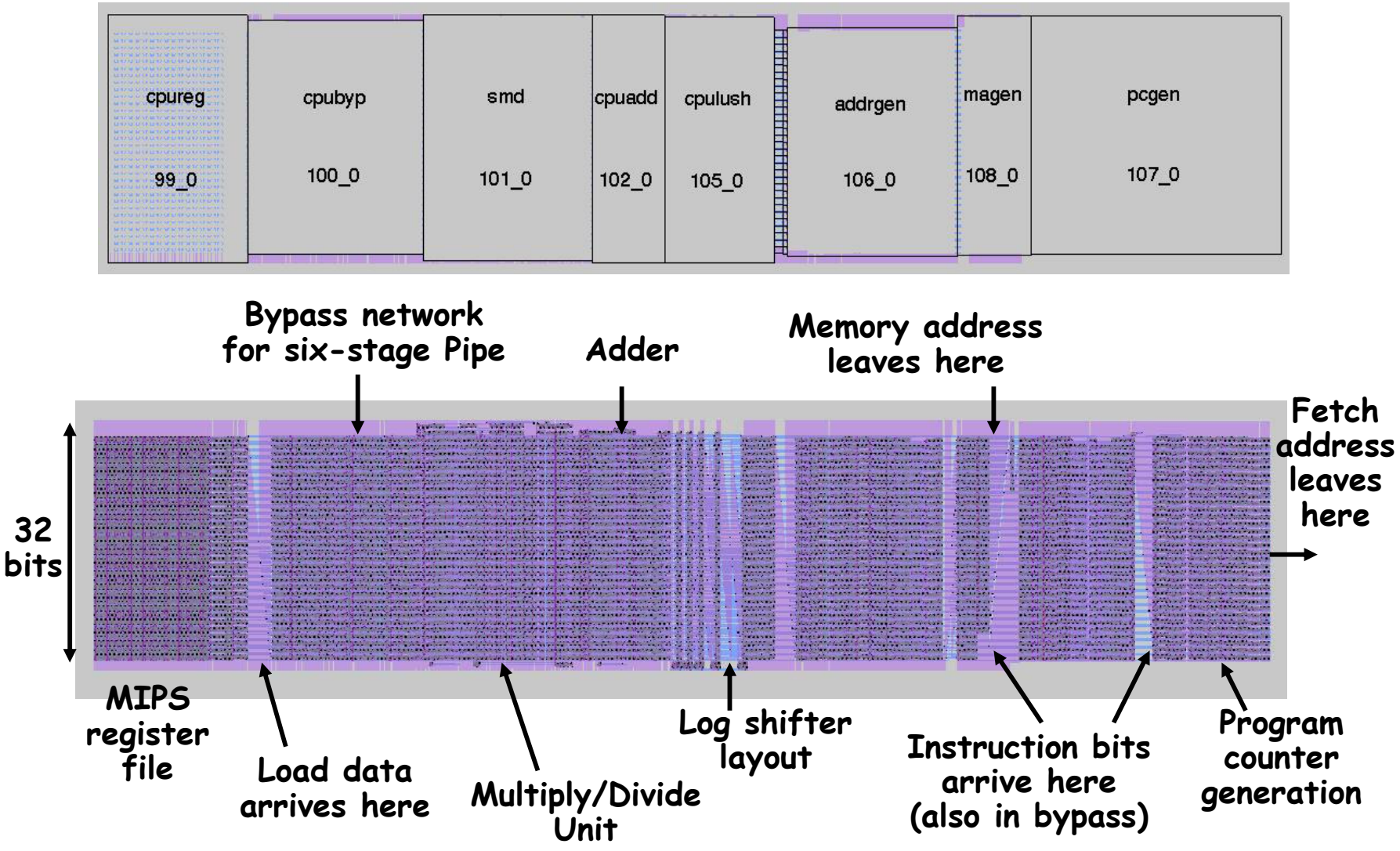
Adapted from [Weste'11]

Custom Datapaths: MIPS Datapath Example (1)



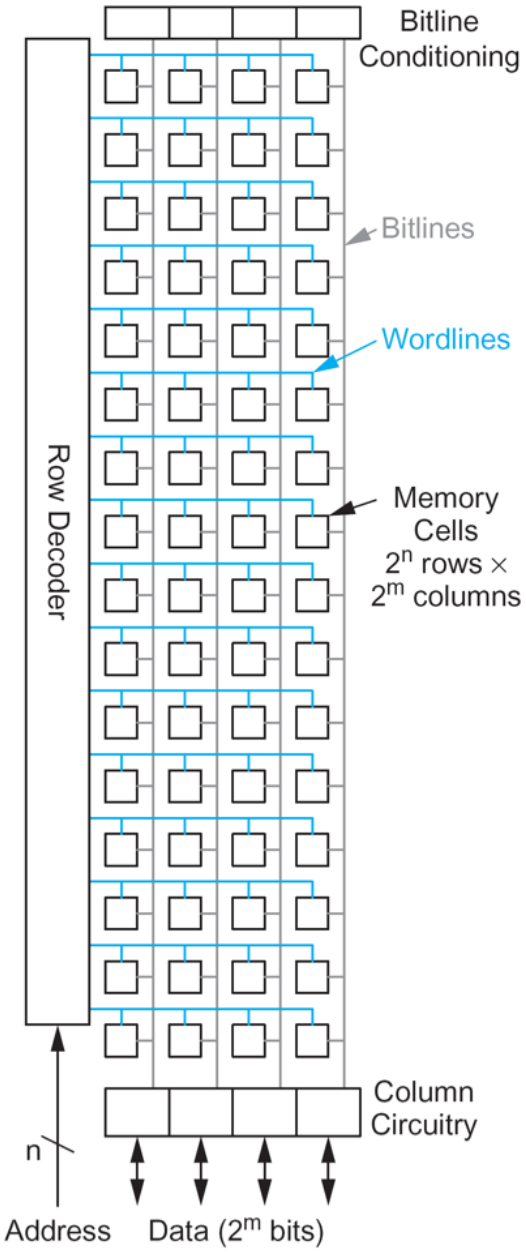
Adapted from [Weste'11]

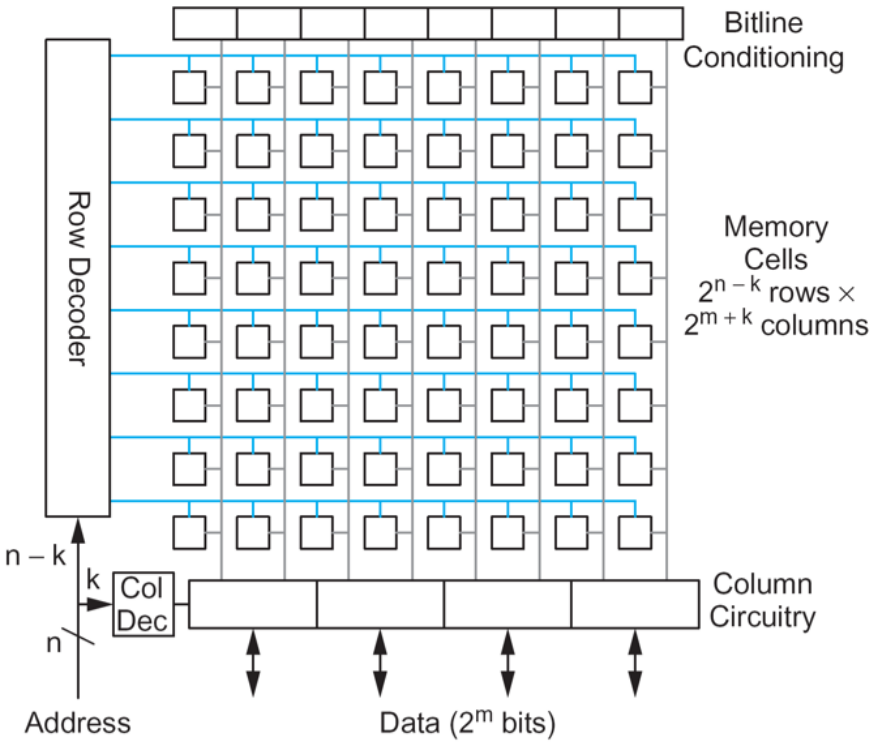
Custom Datapaths: MIPS Datapath Example (2)



Adapted from [Terman'02]

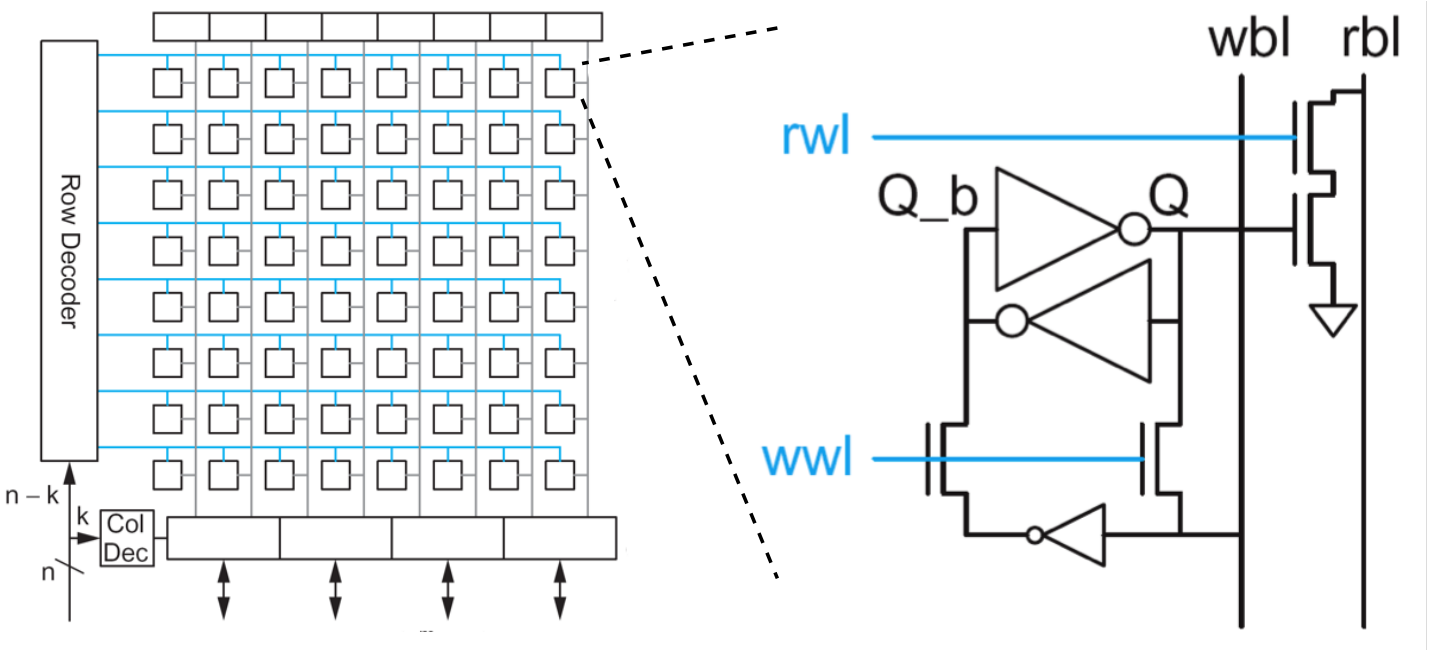
Custom Memories: Array Structure





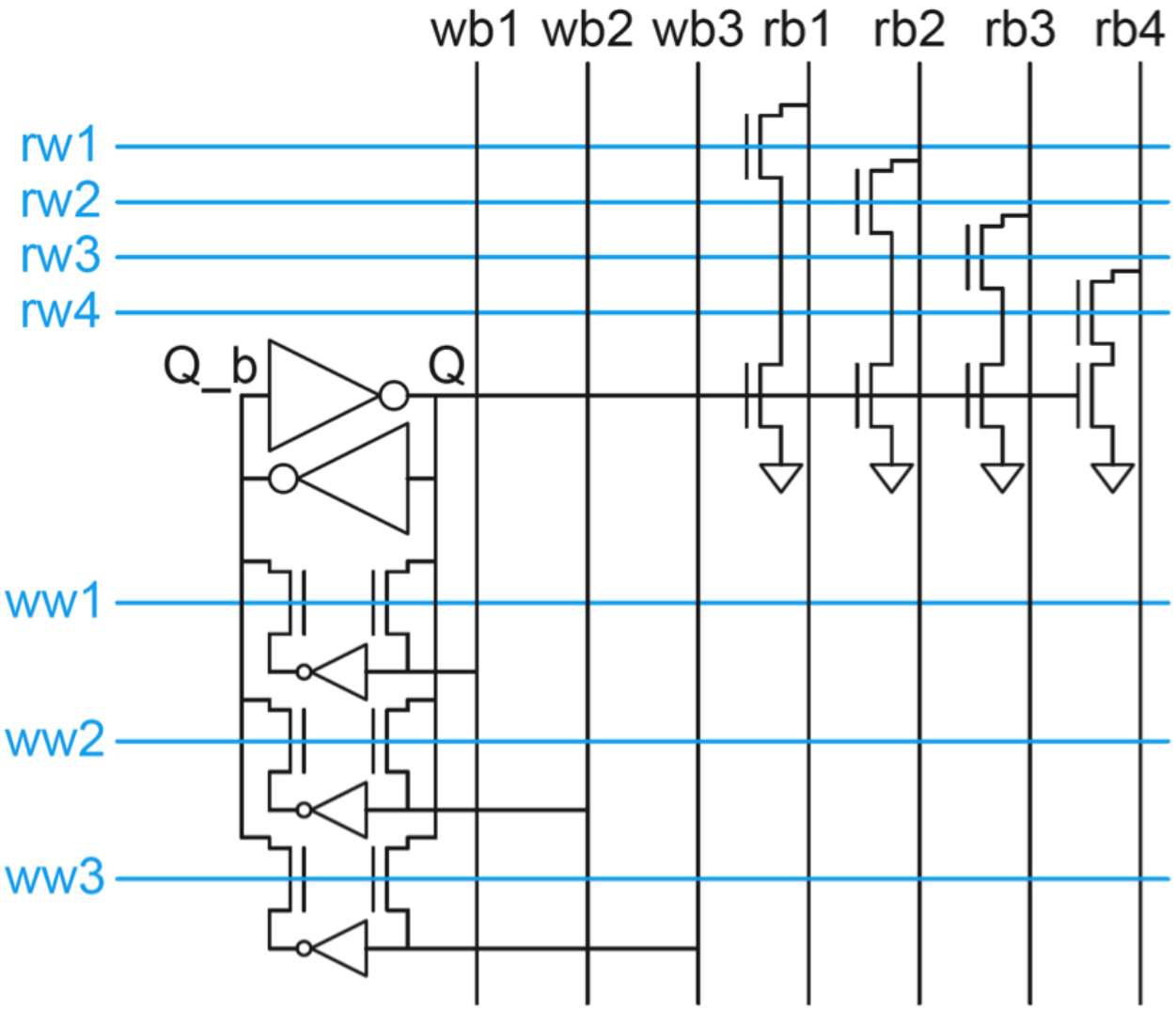
Adapted from [Weste'11]

Custom Memories: Register File Circuits



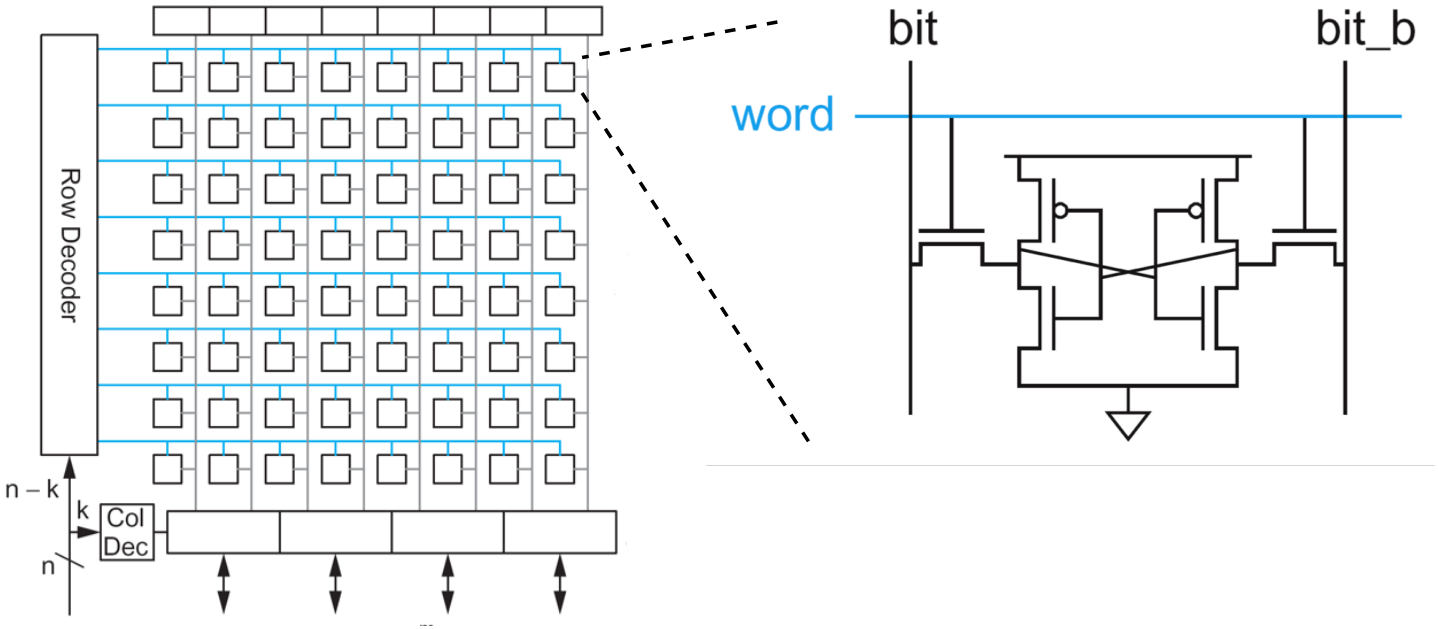
Adapted from [Weste'11]

Custom Memories: Register File Circuits



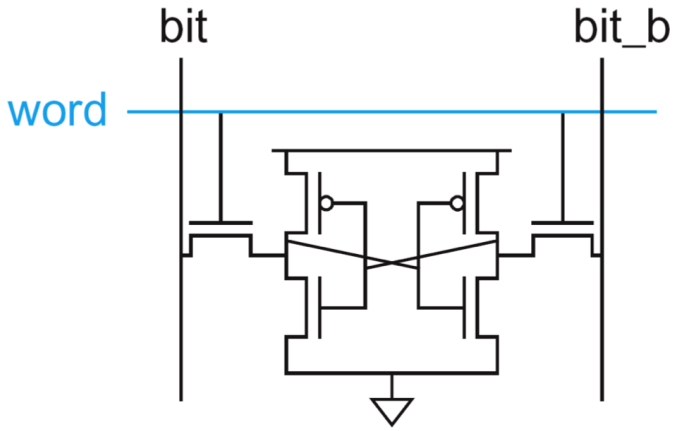
Adapted from [Weste'11]

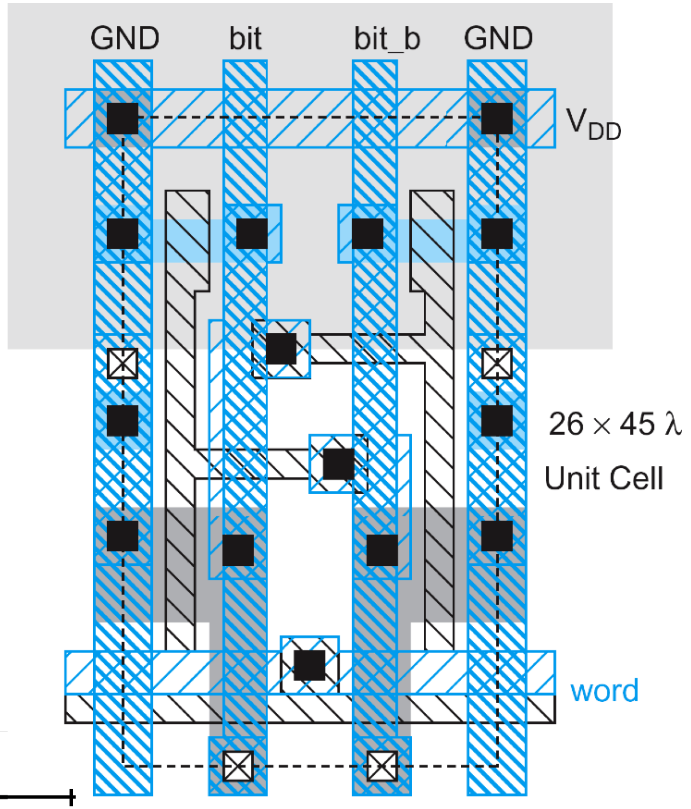
Custom Memories: SRAM Circuits

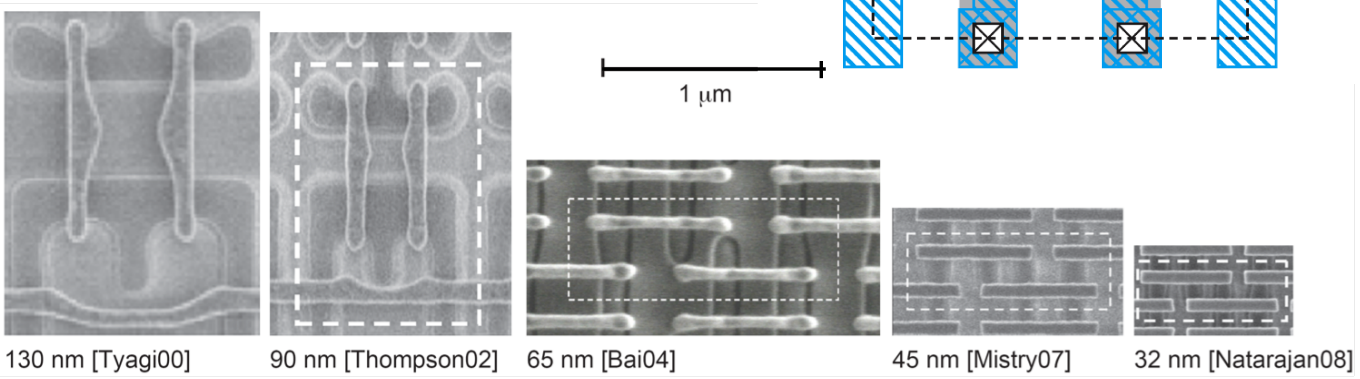


Adapted from [Weste'11]

Custom Memories: SRAM Layut



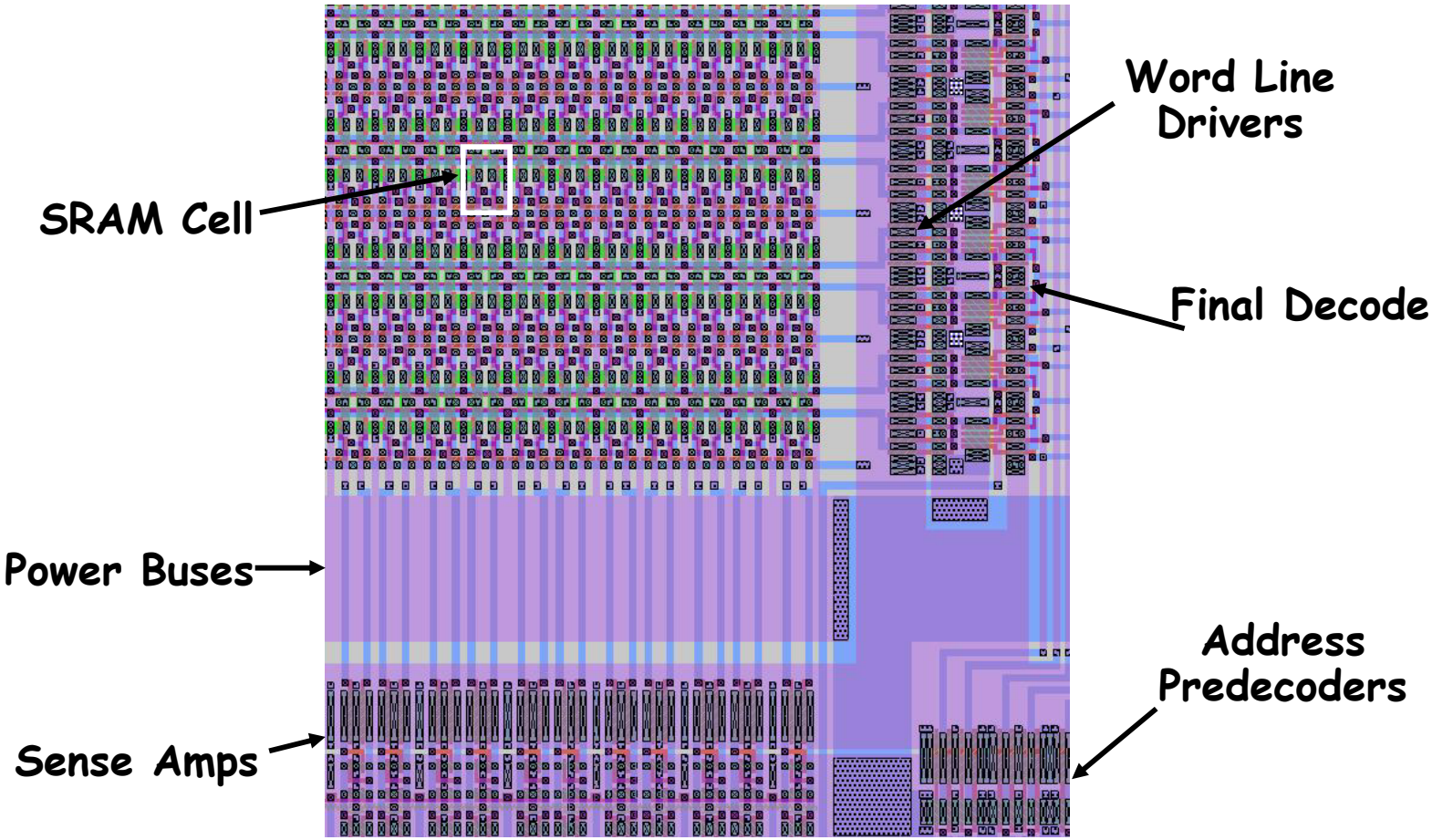




Adapted from [Weste'11]

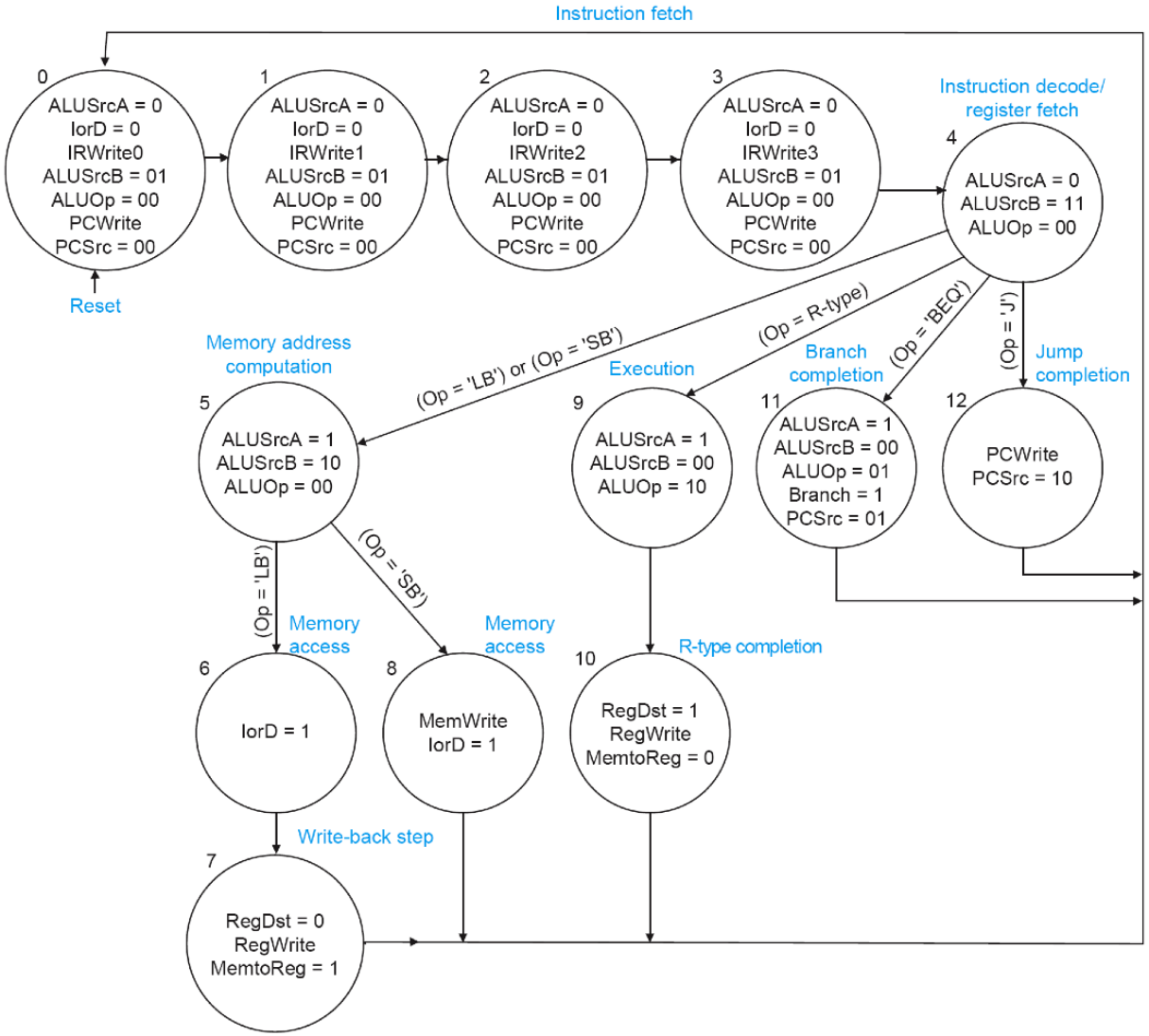
Custom Memories: SRAM

- Regular arrays built with cells that abut in two dimensions
  - Have to pitch match in both dimensions



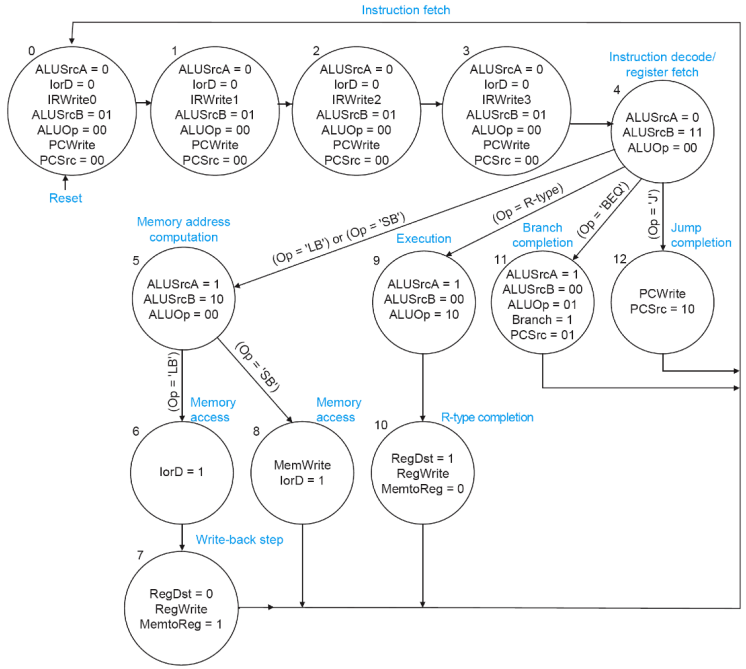
Adapted from [Terman'02]

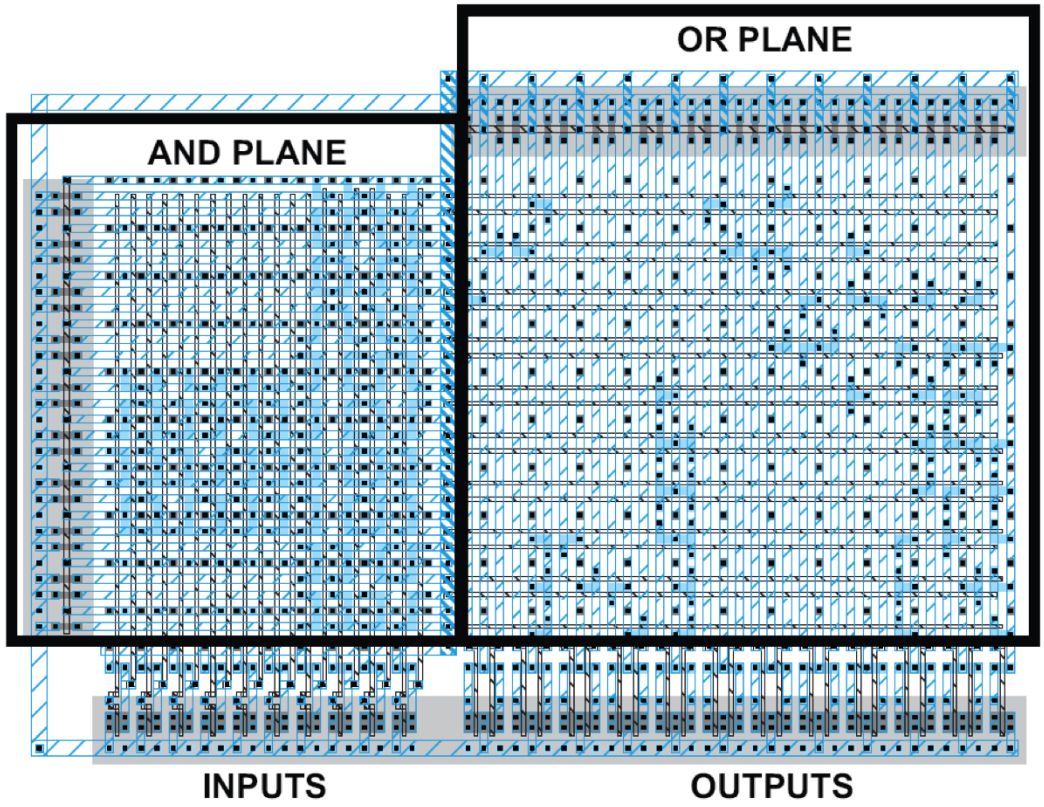
Finite-State-Machine Control Unit



Adapted from [Weste'11]

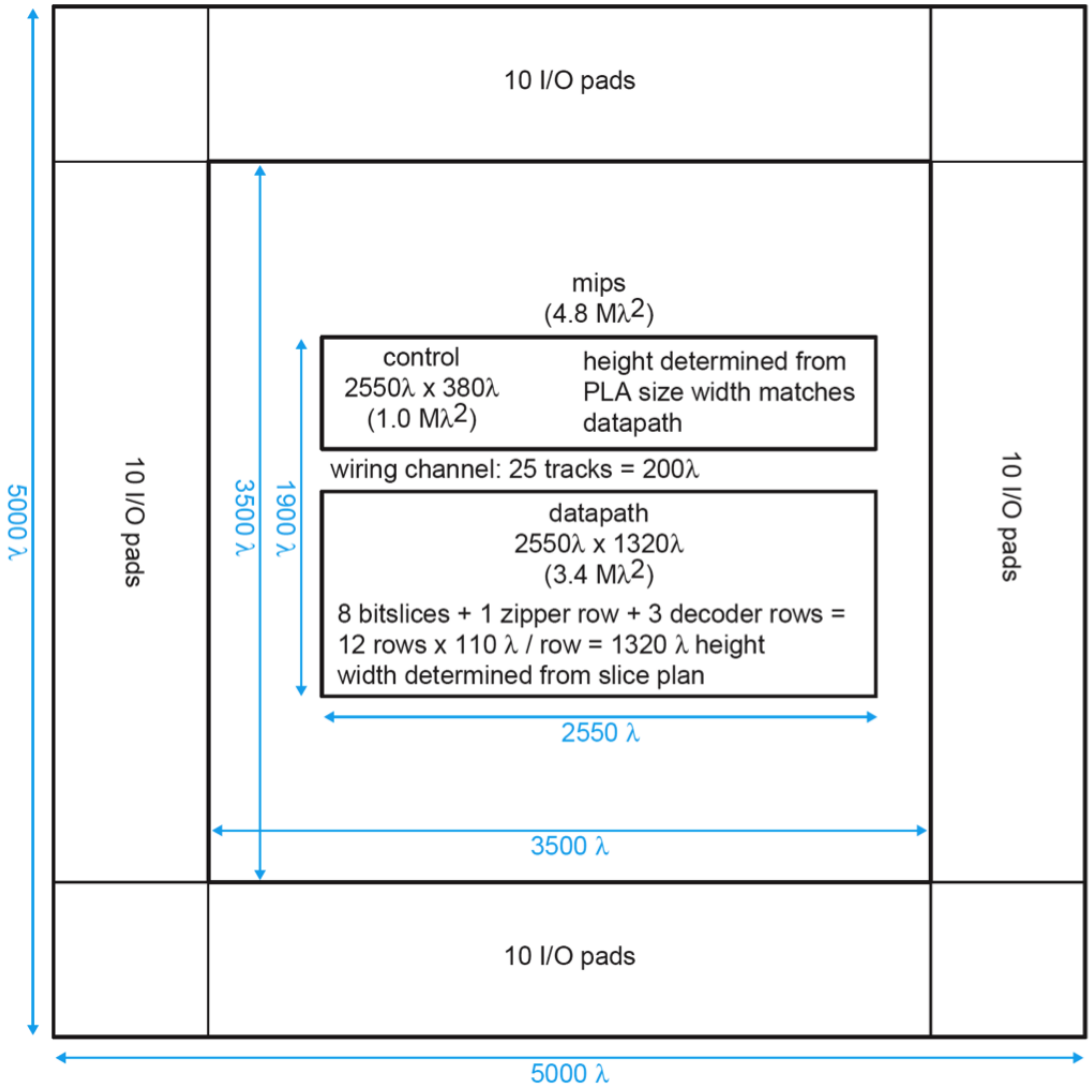
Full Custom Control Logic with PLA





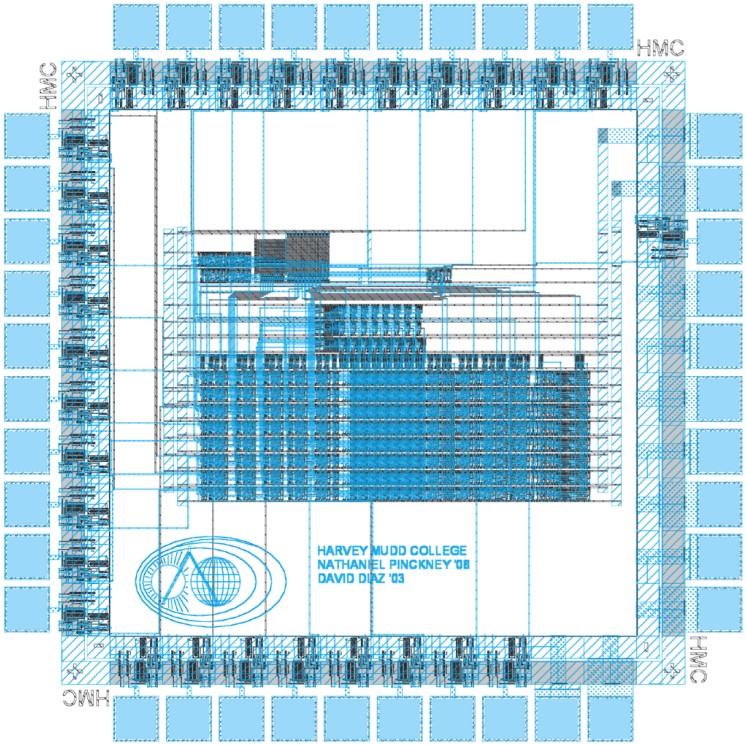
Adapted from [Weste'11]

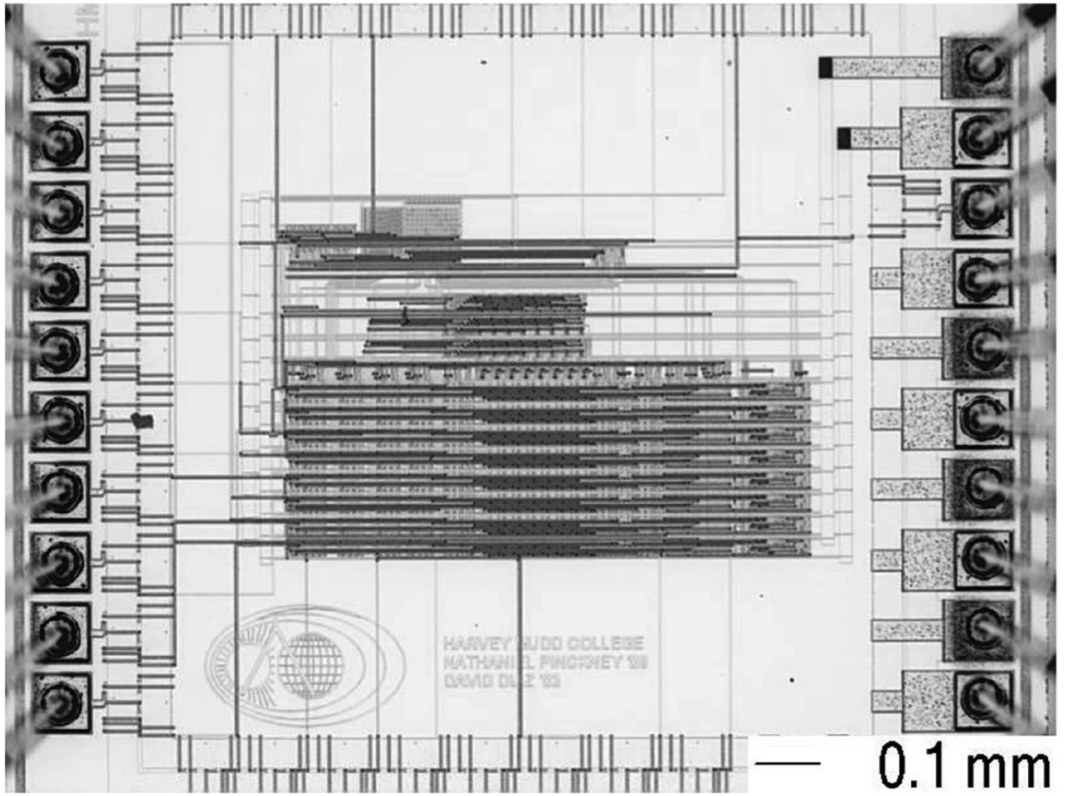
Top-Level Chip Floorplan



Adapted from [Weste'11]

Final Full-Custom MIPS Processor





Adapted from [Weste'11]

Acknowledgments

- [Weste'11] N. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 4th ed, Addison Wesley, 2011.
- [Terman'02] C. Terman and K. Asanović, MIT 6.371 Introduction to VLSI Systems, Lecture Slides, 2002.
- [Ellervee'04] P. Ellervee, IAY3714 VLSI Synthesis and HDLs, Lecture Slides, 2004.