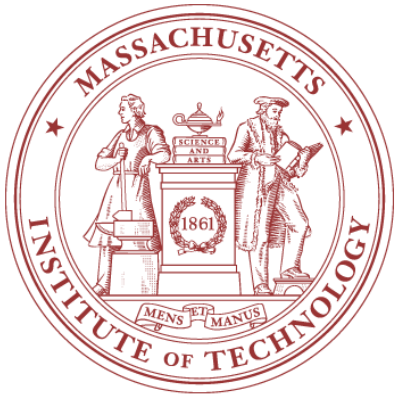
Application Specific Integrated Circuit Design

Lecture 5

Vladimir Stojanović



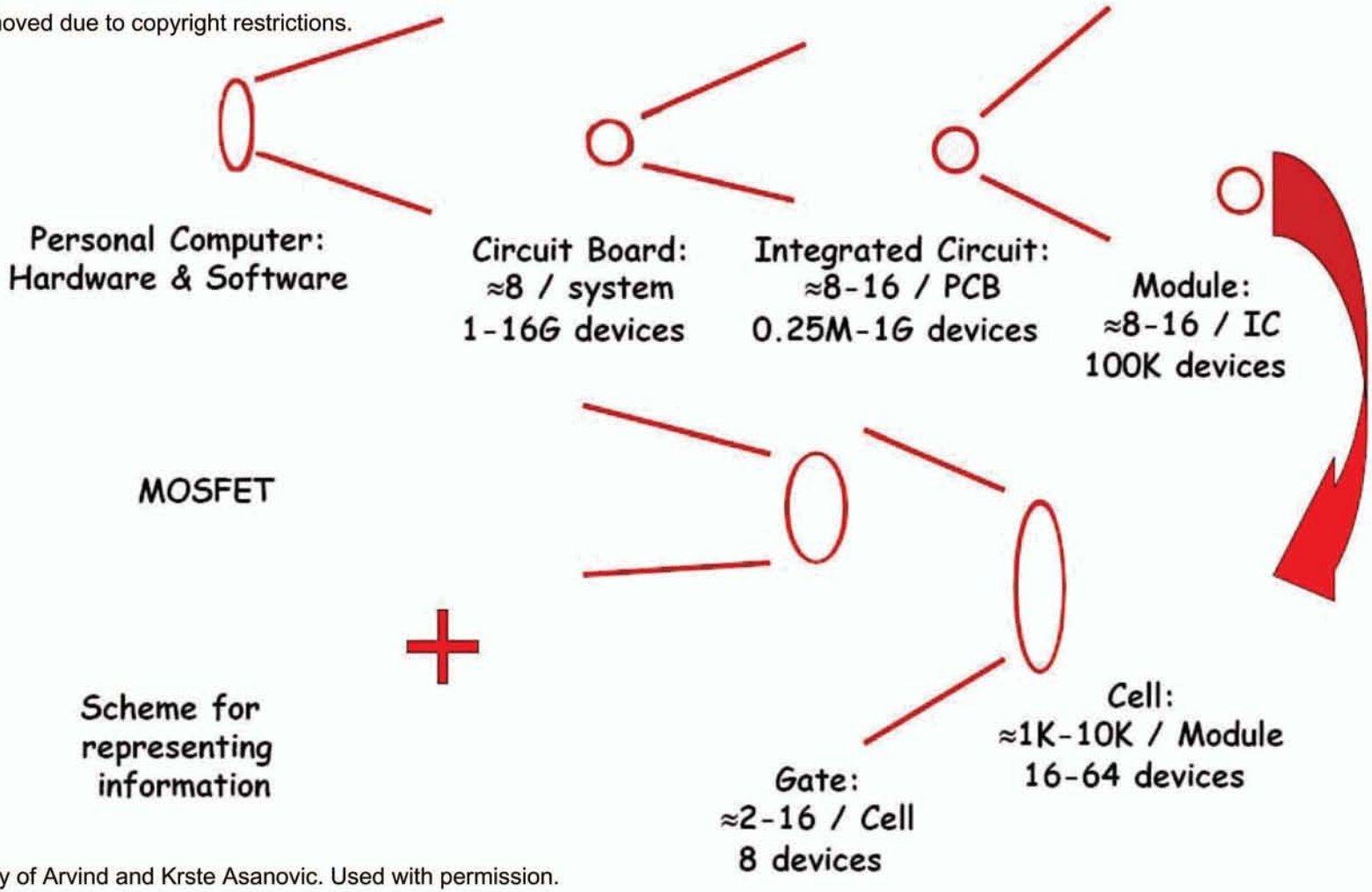
6.973 Communication System Design – Spring 2006

Massachusetts Institute of Technology

Modern digital systems engineering

- Managing complexity and connectivity

Images removed due to copyright restrictions.



Courtesy of Arvind and Krste Asanovic. Used with permission.

Chip design styles

- Full-custom
  - Transistors are hand-drawn
  - Best performance (although almost extinct)
    - Alpha processors, older Intel processors
  - Recent processors are semi-custom (Sun, AMD, Intel)
- Standard-Cell-based ASICs
  - Only use standard cells from the library
  - Dominant design style for non-processor, comms and multimedia ASICs
  - This is what we will use in 6.973 (also used in 6.375)
- Cheaper alternatives (for small volumes)
  - Sea of Gates (mask-programmable gate arrays)
  - Field Programmable Gate Arrays (FPGA)
    - On-the fly reconfigurable interconnect
- Flexibility vs. cost
  - Tighter control over transistors increases design cost
    - Can make faster designs but harder to verify and more expensive

Modern Application-Specific IC (ASIC)

- Multiple functional blocks
  - Put together at the top level
    - Makes a WLAN chip
  - Lots of modeling
    - Behavioral level
    - Architecture/functional level
  - Different teams on each block
    - Need to make sure things work
      - When connected
  - Many levels of hierarchy
    - Lots of iteration and reuse
- Many architectural choices
  - Use lots of implementation tricks
    - Micro-architecture and algorithmic transforms
    - Straightforward solutions many times the chip size
- Sophisticated CAD tools to architect and verify 4M design

Our ASIC example:
802.11a PHY

Image removed due to copyright restrictions.

[Thompson02]

Macro modules

256×32 (or 8192 bit) SRAM Generated by hard-macro module generator

Image removed due to copyright restrictions.

- **Generate highly regular structures (entire memories, multipliers, etc.) with a few lines of code**
- **Verilog models for memories automatically generated based on size**

- Example – chip-in-a-day flow (B. Brodersen, UC Berkely)
  - A bunch of macros pre-generated (multipliers, adders, memories)
  - Easy to do comm system design

Courtesy of Anantha Chandrakasan. Used with permission.

Gate Arrays

Can cut mask costs by prefabricating arrays of transistors on wafers

Only customize metal layer for each design

- Fixed-size unit transistors
- Metal connections personalize design

Two kinds:

- Channeled Gate Arrays
  - Leave space between rows of transistors for routing
- Sea-of-Gates
  - Route over the top of unused transistors

Image removed due to copyright restrictions.

Courtesy of Arvind and Krste Asanovic. Used with permission.

[OCEAN Sea-of-Gates Base Pattern]

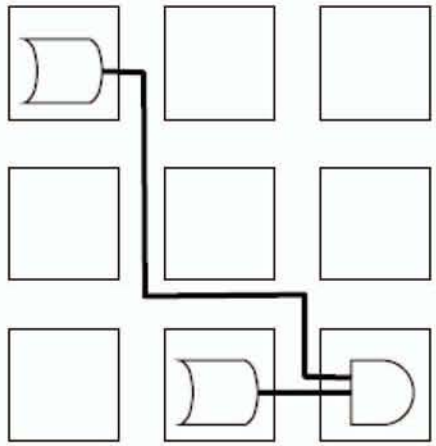
Gate Array Pros and Cons

- **Cheaper and quicker since less masks to make**
  - Can stockpile wafers with diffusion and poly finished
- **Memory inefficient when made from gate array**
  - Embedded gate arrays add multiple fixed memory blocks to improve density (=>Structured ASICs)
  - Cell-based array designed to provide efficient memory cell (6 transistors in basic cell)
- **Logic slow and big due to fixed transistors and wiring overhead**
  - Advanced cell-based arrays hardwire logic functions (NANDs/NORs/LUTs) which are personalized with metal

Field-Programmable Gate Arrays (FPGA)

- Each cell in array contains a programmable logic function
- Array has programmable interconnect between logic functions
- Arrays mass-produced and programmed by customer after fabrication
  - Can be programmed by blowing fuses, loading SRAM bits, or loading FLASH memory
- Overhead of programmability makes arrays expensive and slow but startup costs are low, so much cheaper than ASIC for small volumes

Image removed due to copyright restrictions.

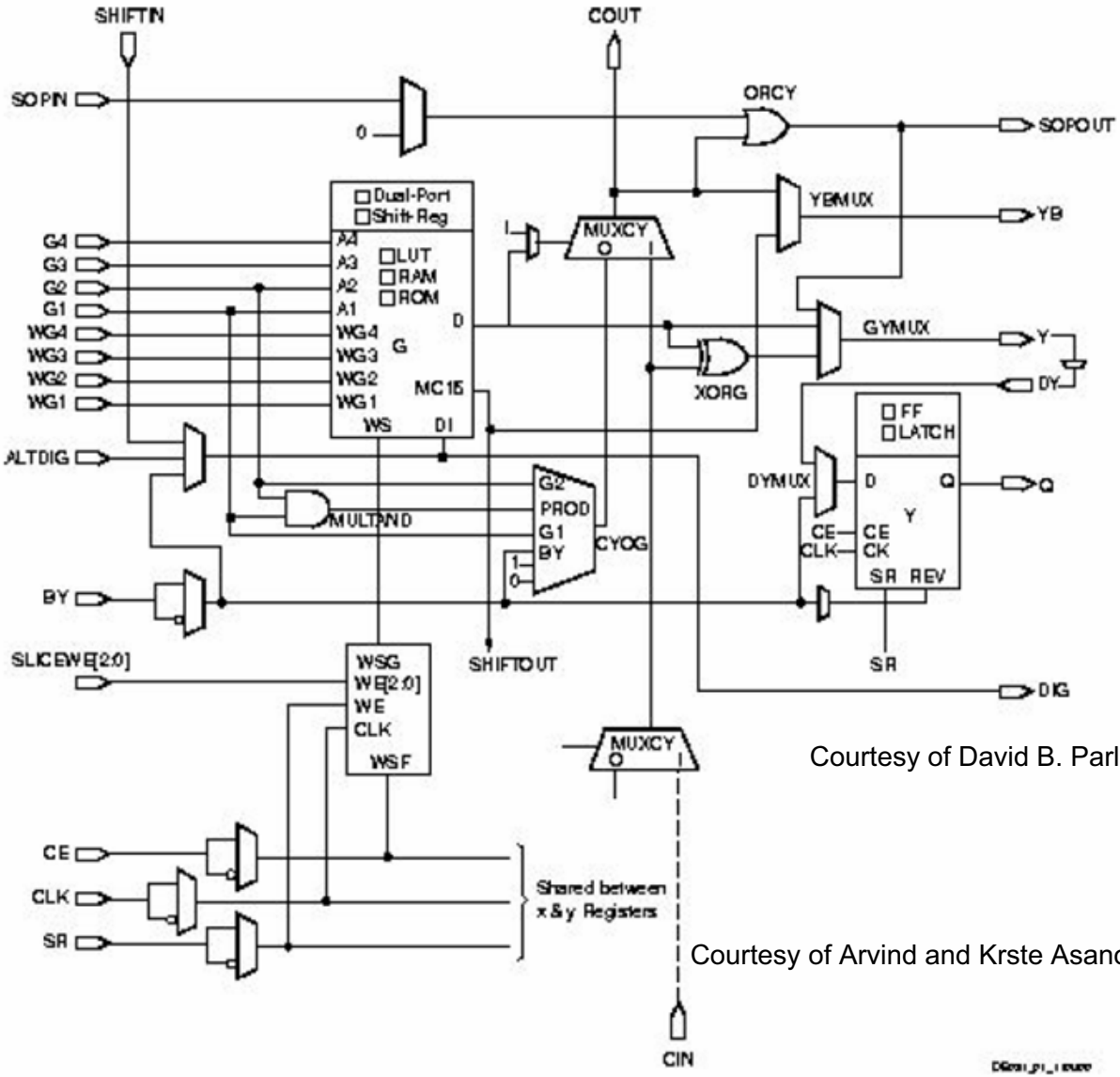


Virtex4 FPGA

Courtesy of Arvind and Krste Asanovic. Used with permission.

Xilinx Configurable logic block (CLB)

- Full-add, LUT, shift, mux, xor, ff

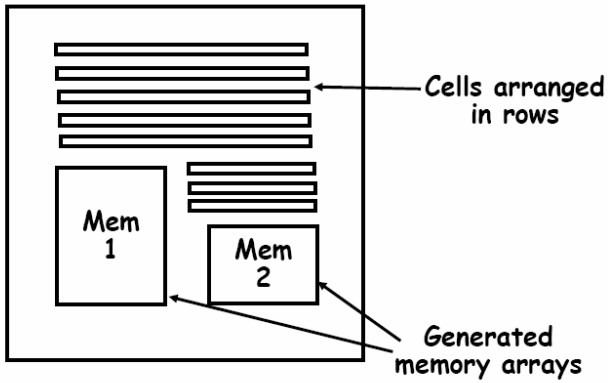


Courtesy of David B. Parlour.

Courtesy of Arvind and Krste Asanovic. Used with permission.

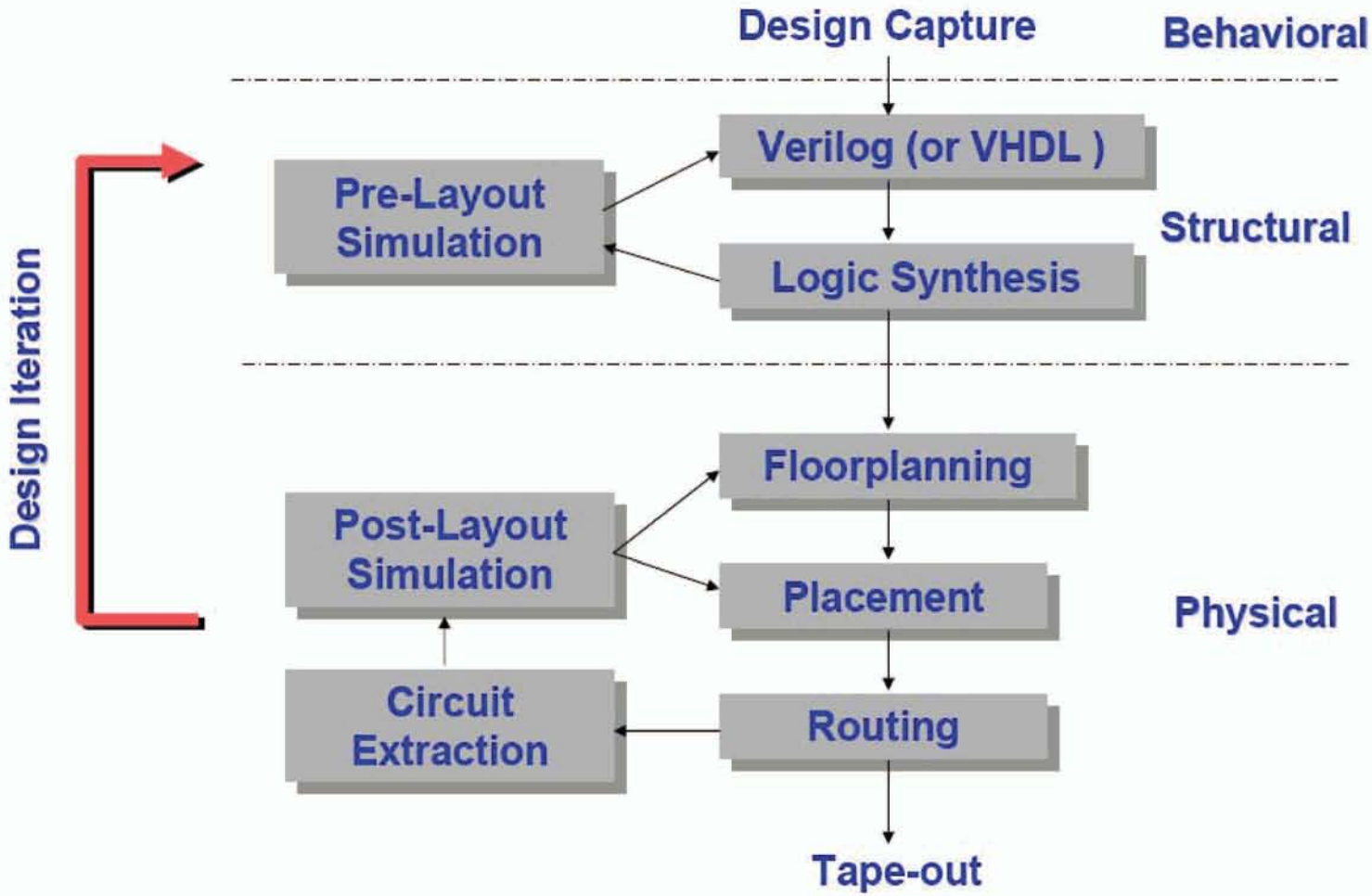
Standard cell ASICs

- Also called Cell-Based ICs (CBICs)
- Fixed library of cells
  - Memory generators
- Design
  - Cells synthesized from hardware descriptive language (Verilog, VHDL)
  - Cells manually entered in a schematics
  - Placed and routed automatically (most desirable)
- Full set of masks for productions
  - Most popular today, but increasingly expensive due to mask costs in advanced technologies
  - Currently a mask set is a couple $M
    - FPGAs are getting increased attention
- We will use this approach



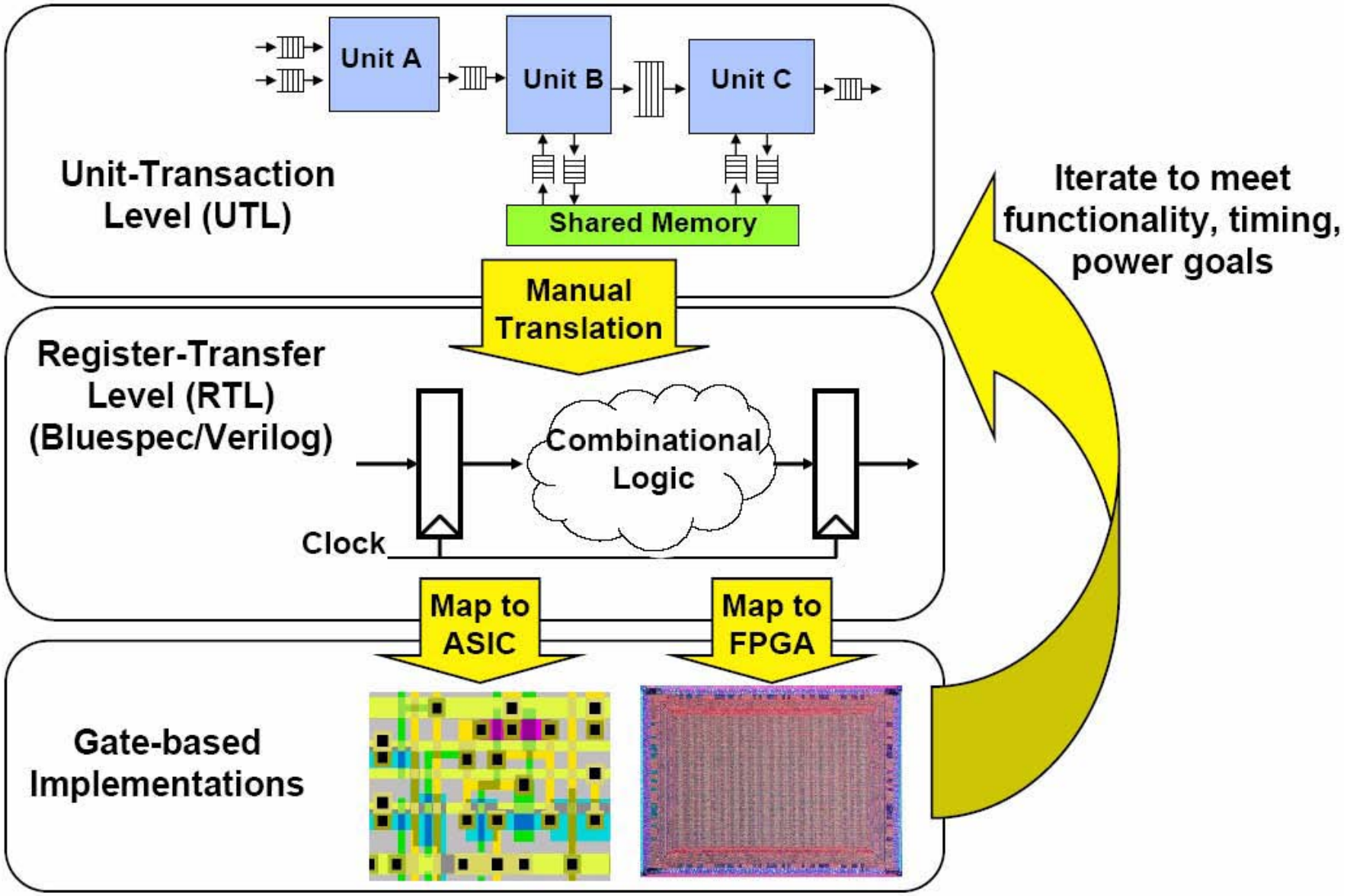
Courtesy of Arvind and Krste Asanovic. Used with permission.

The ASIC flow



Most Common Design Approach for Designs up to 500Mhz Clock Rates

ASIC flow subset – 6.375 and 6.973



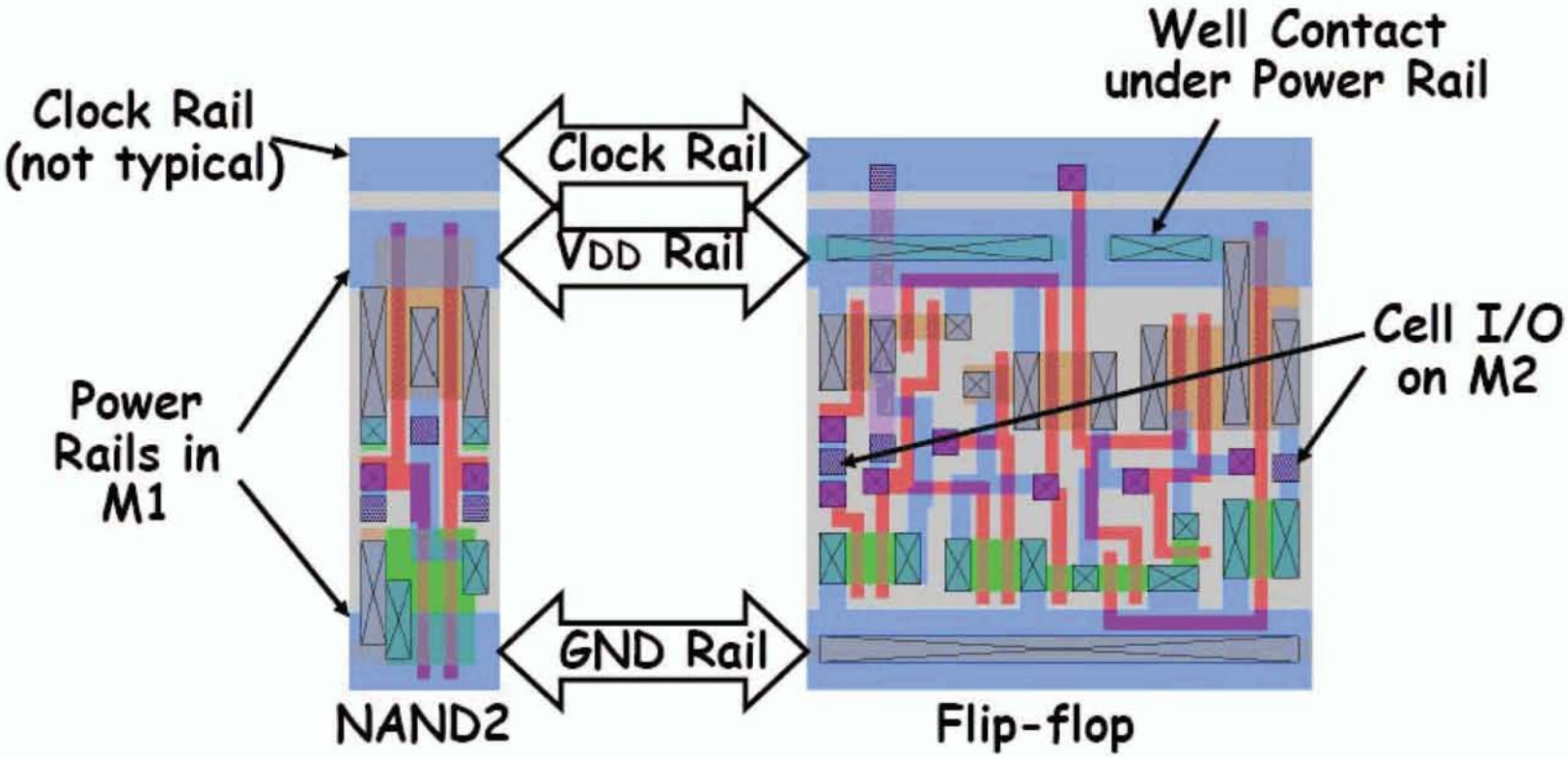
Courtesy of Arvind and Krste Asanovic. Used with permission.

Standard cell design

- Layout considerations

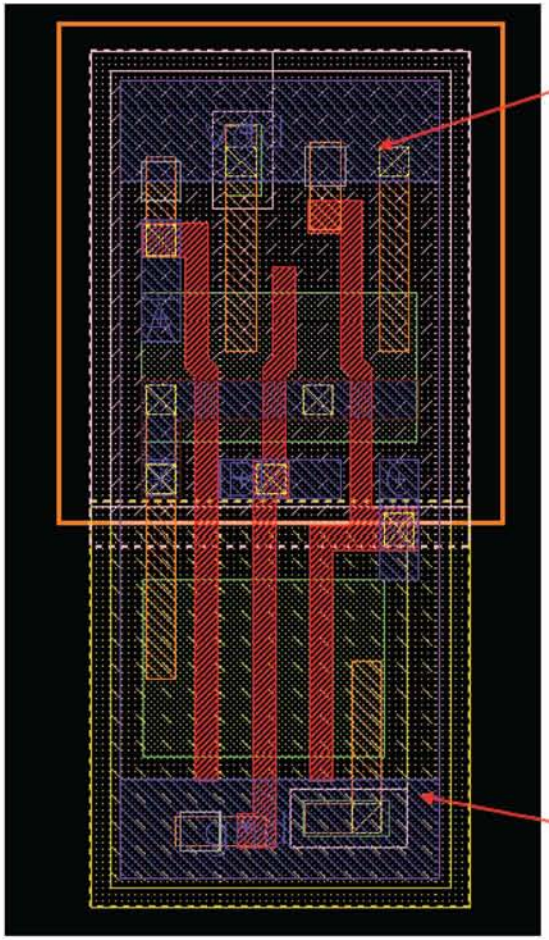
  Cells have standard height but vary in width

  Designed to connect power, ground, and wells by abutment



Courtesy of Arvind and Krste Asanovic. Used with permission.

Standard cell characterization



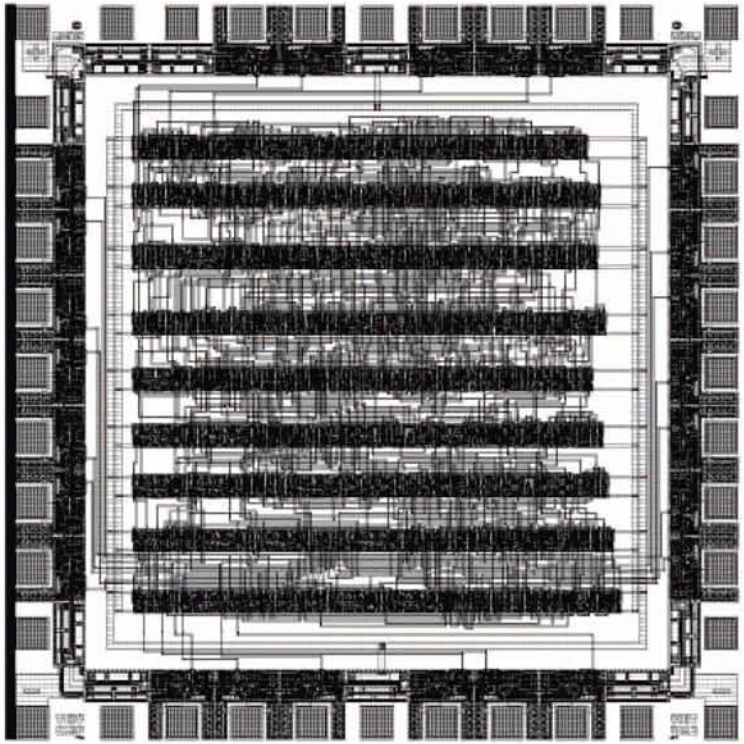
Path	1.2V - 125°C	1.6V - 40°C
$In1—t_{pLH}$	$0.073+7.98C+0.317T$	$0.020+2.73C+0.253T$
$In1—t_{pHL}$	$0.069+8.43C+0.364T$	$0.018+2.14C+0.292T$
$In2—t_{pLH}$	$0.101+7.97C+0.318T$	$0.026+2.38C+0.255T$
$In2—t_{pHL}$	$0.097+8.42C+0.325T$	$0.023+2.14C+0.269T$
$In3—t_{pLH}$	$0.120+8.00C+0.318T$	$0.031+2.37C+0.258T$
$In3—t_{pHL}$	$0.110+8.41C+0.280T$	$0.027+2.15C+0.223T$

3-input NAND cell
(from ST Microelectronics):
C = Load capacitance
T = input rise/fall time

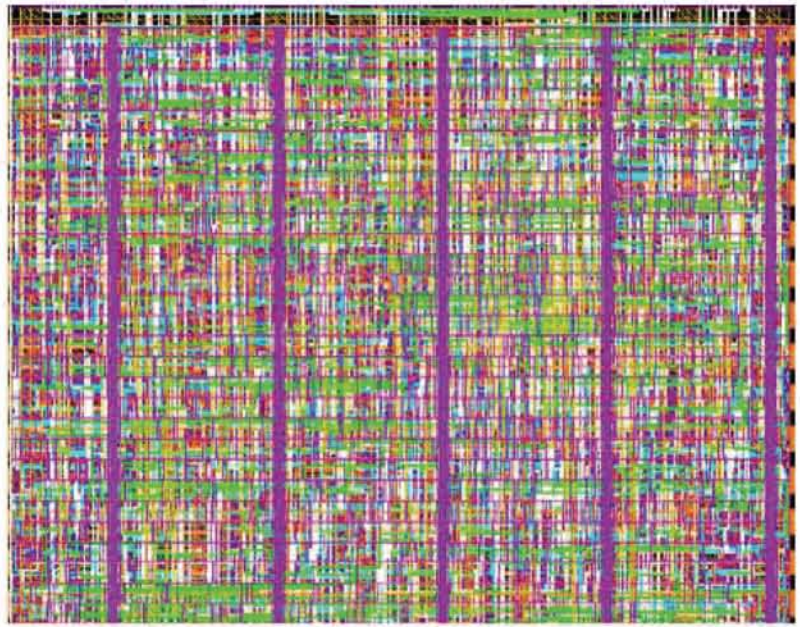
- **Each library cell (FF, NAND, NOR, INV, etc.) and the variations on size (strength of the gate) is fully characterized across temperature, loading, etc.**

Standard cell layout methodology

2-level metal technology



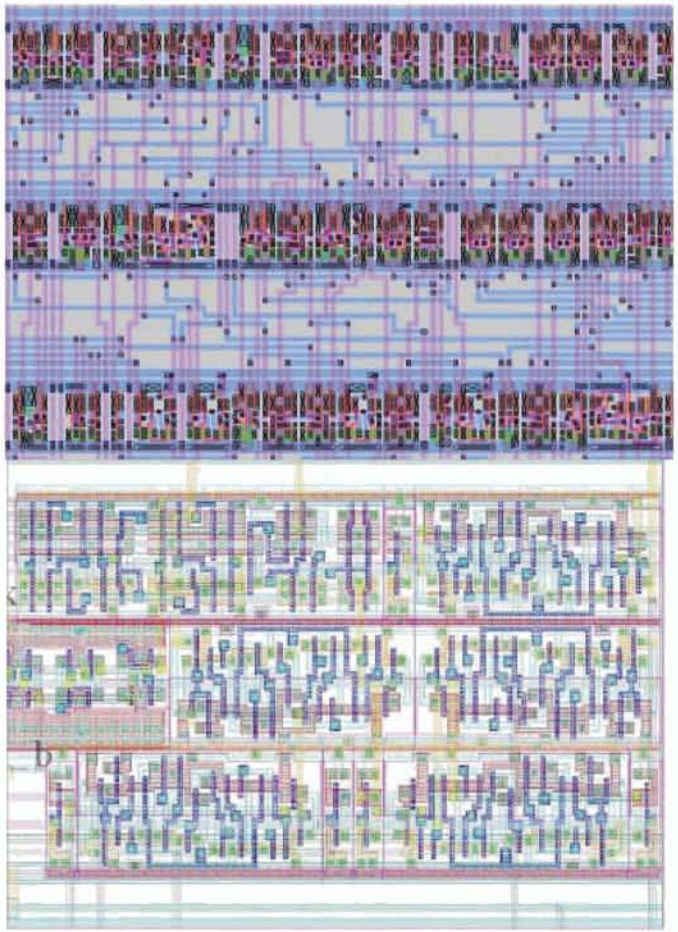
Current Day Technology



Cell-structure hidden under interconnect layers

- **With limited interconnect layers, dedicated routing channels between rows of standard cells are needed**
- **Width of the cell allowed to vary to accommodate complexity**
- **Interconnect plays a significant role in speed of a digital circuit**

More standard cell layouts



Channel routing for
1.0μm 2-metal stdcells

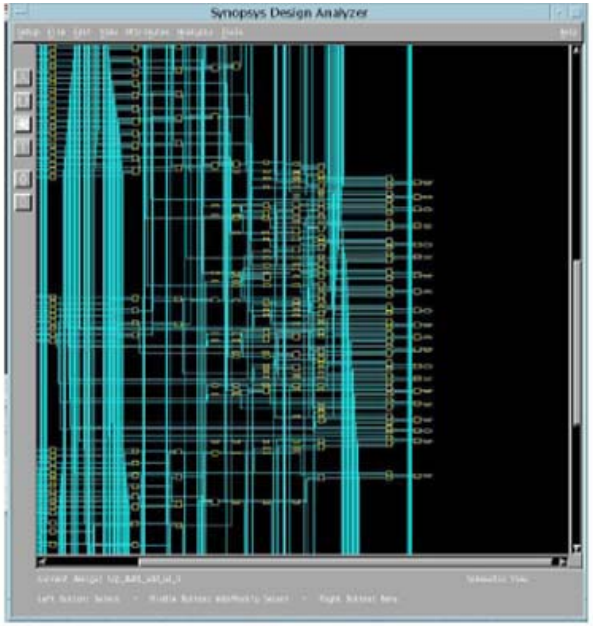
Over cell routing for
0.18μm 6-metal stdcells

The front-end: Verilog to ASIC layout flow

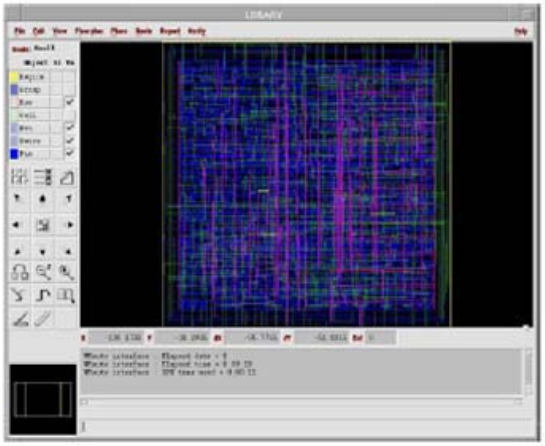
- The “push-button” approach

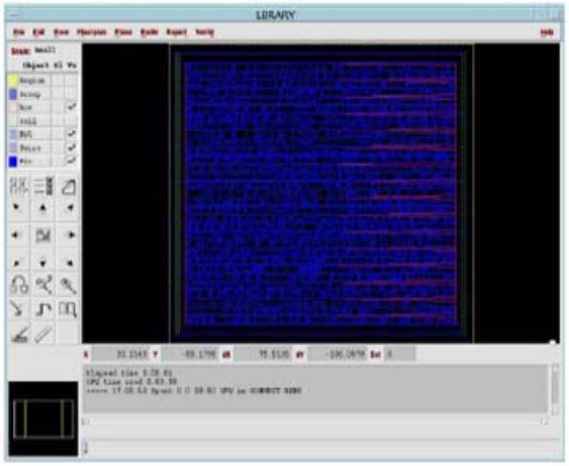
```
module adder64 (a, b, sum);
  input  [63:0] a, b;
  output [63:0] sum;

  assign sum = a + b;
endmodule
```

After Synthesis



After Routing



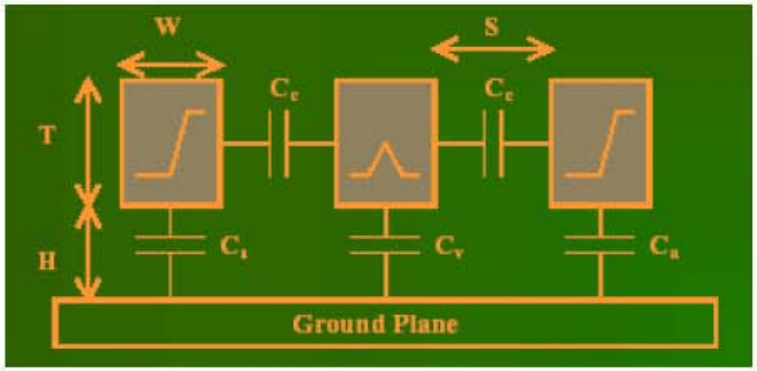


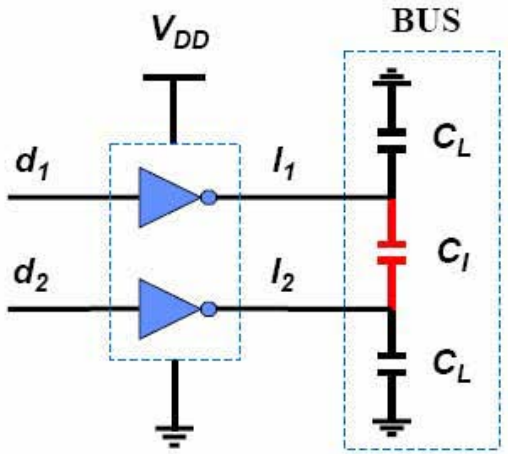
After Placement

Courtesy of Anantha Chandrakasan. Used with permission.

The back-end: The "Design closure" problem

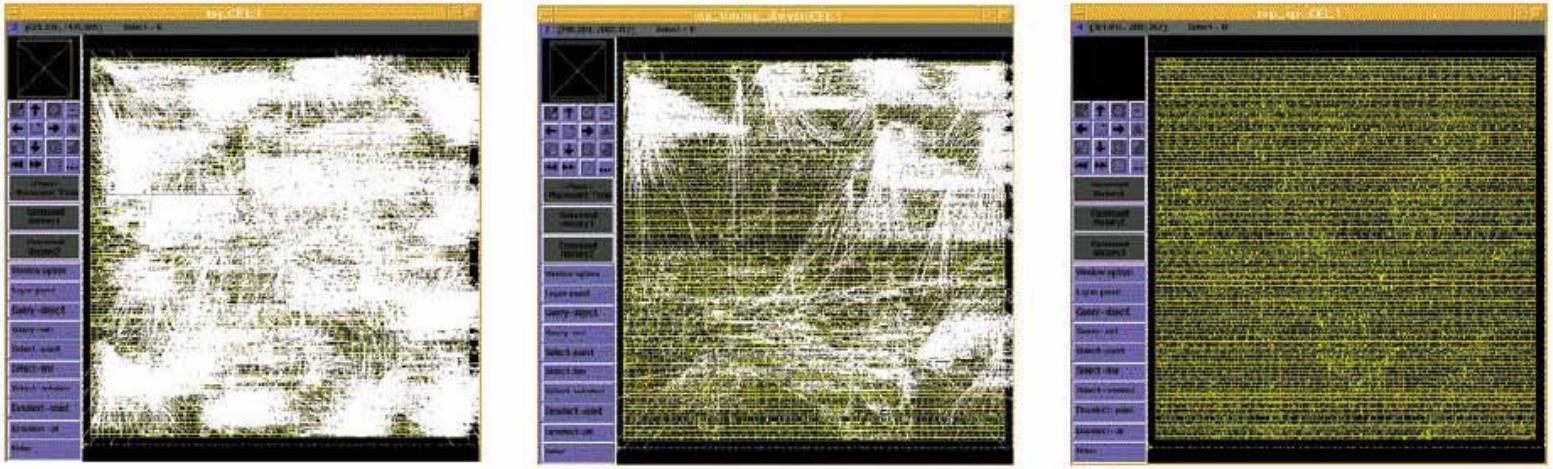
- Biggest problem are wires (signals and clock)





$$\lambda = \frac{C_I}{C_L} = 5$$

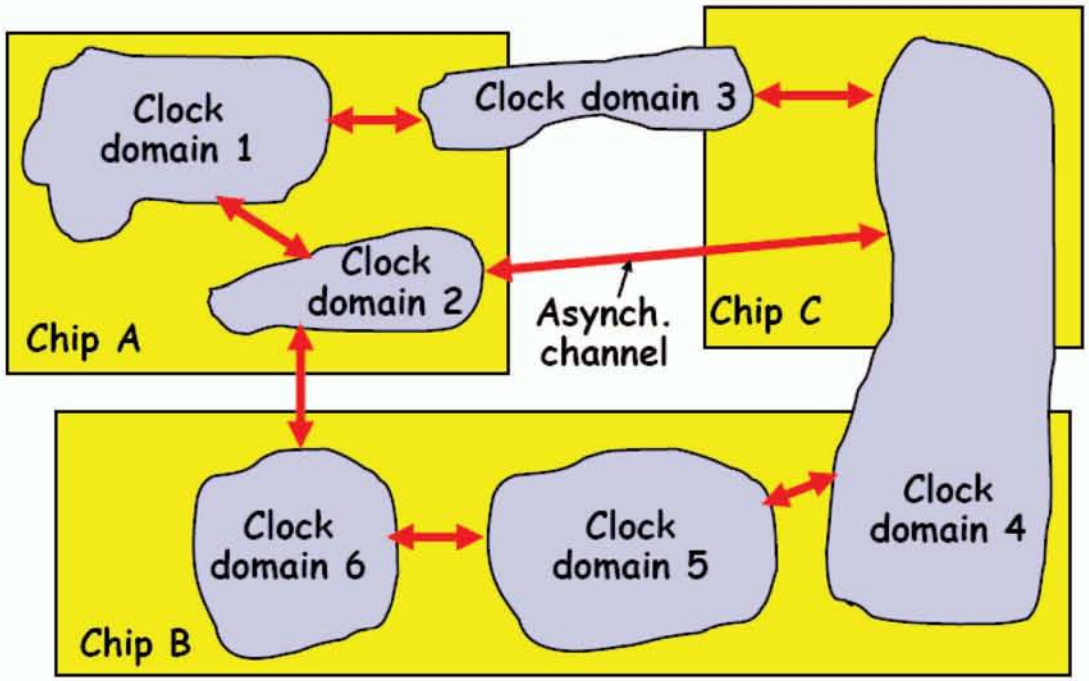
Wire-to-wire capacitance causes inter-wire delay dependencies



Iterative Removal of Timing Violations (white lines)

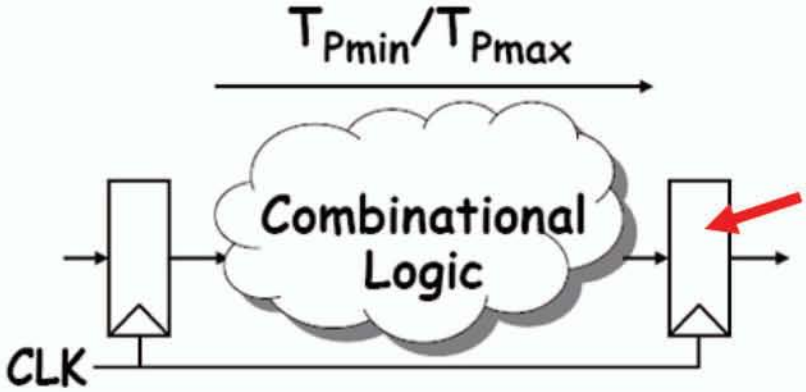
Large Systems

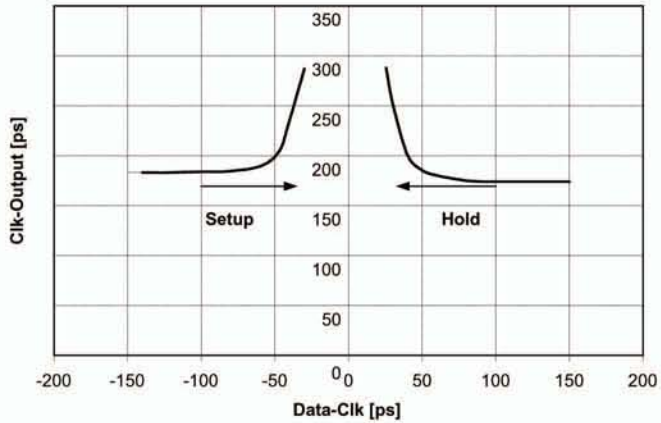
- Most large scale ASICs, and systems built with these ASICs, have several synchronous clock domains connected by asynchronous communication channels



- We'll focus on a single synchronous clock domain today

Single Clock Edge-Triggered Design





Single clock with edge-triggered registers most common design style in ASICs

- Slow path timing constraint

$$T_{cycle} \geq T_{CQmax} + T_{Pmax} + T_{setup}$$

  - can always work around slow path by using slower clock

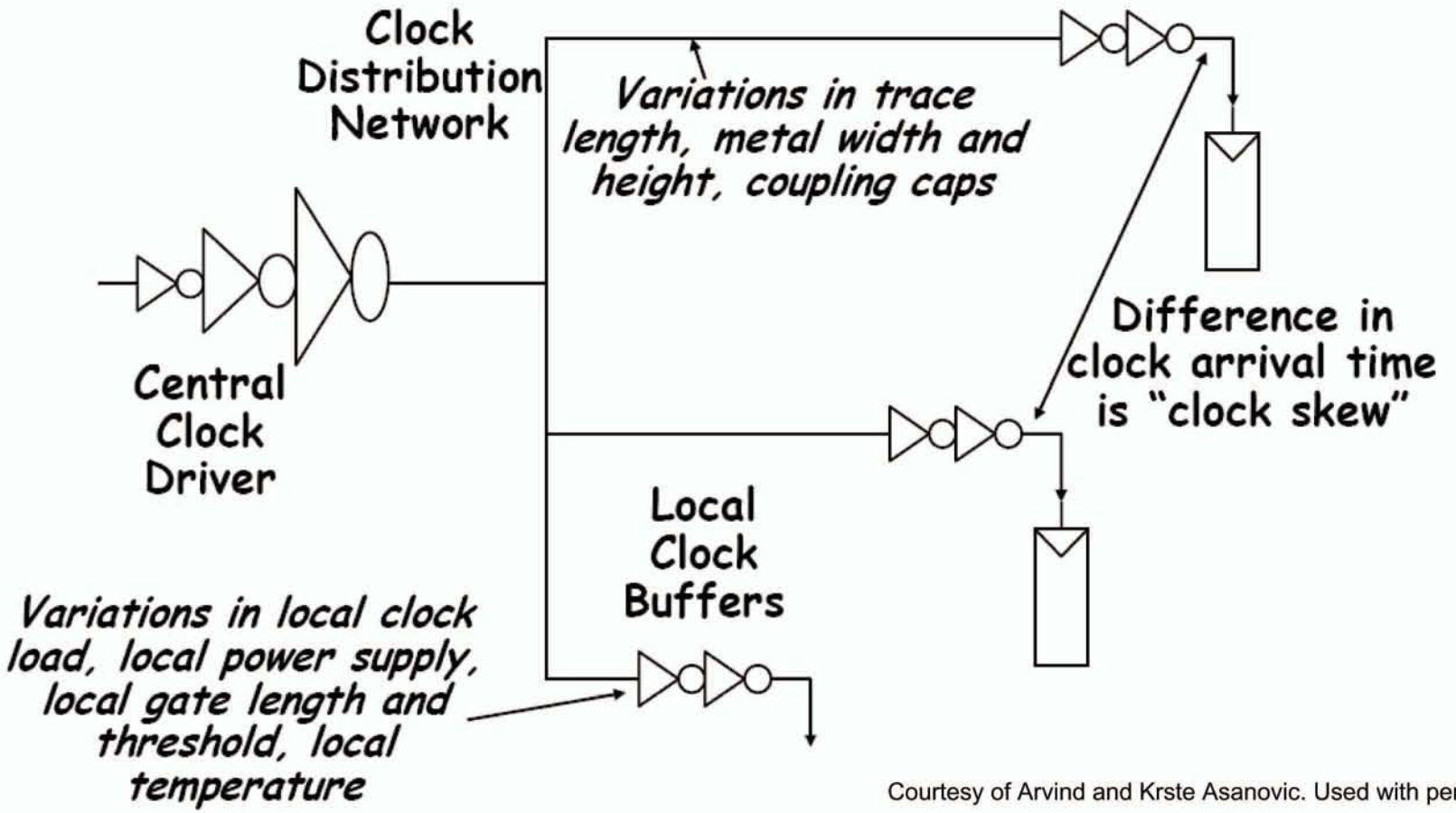
- Fast path timing constraint

$$T_{CQmin} + T_{Pmin} \geq T_{hold}$$

  - bad fast path cannot be fixed without redesign!
  - might have to add delay into paths to satisfy hold time

Clock distribution

- Can't really distribute clock at same instant to all flip-flops on chip



Clock grids

- Alpha 21264 grid example
  - Minimize skew and jitter
  - Costs a lot of power

21064

buffer tree

21164

Images removed due to copyright restrictions.

21264

21064 21164 21264

Clock trees

binary tree

H-tree

Images removed due to copyright restrictions.

X-tree

Arbitrary
matched tree

- Recursive pattern to match delay
  - Much less power than grid
  - More skew and jitter
    - Skew
      - Non-uniform loading
      - Buffer mismatch
    - Jitter
      - Supply noise on buffers

Modern Interconnect Stack

Images removed due to copyright restrictions.

Tungsten local interconnect

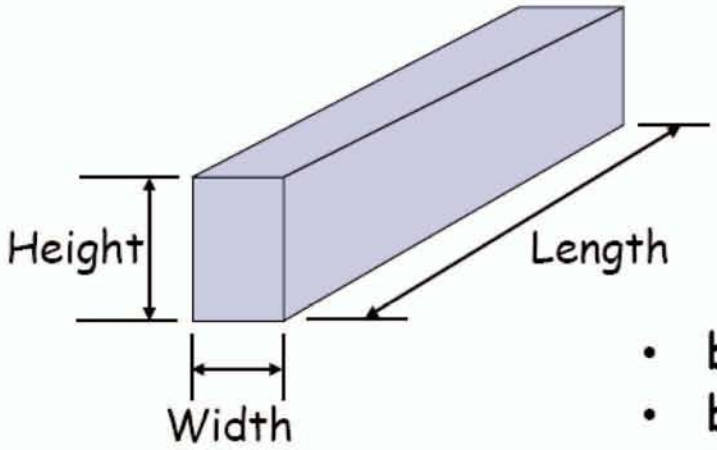
IBM CMOS7 process

6 layers of copper wiring

1 layer of tungsten local interconnect

Courtesy of Arvind and Krste Asanovic. Used with permission.

Wire resistance

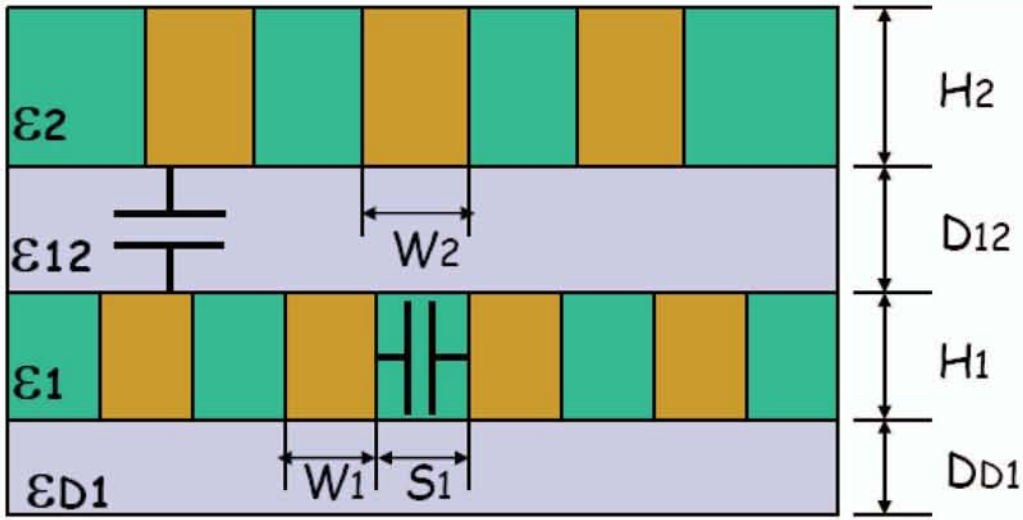


$$\text{resistance} = \frac{(\text{length} \times \text{resistivity})}{(\text{height} \times \text{width})}$$

- bulk aluminum $2.8 \times 10^{-8}\ \Omega$-m
- bulk copper $1.7 \times 10^{-8}\ \Omega$-m
- bulk silver $1.6 \times 10^{-8}\ \Omega$-m

- Height (Thickness) fixed in given manufacturing process
- Resistances quoted as Ω/square
- TSMC 0.18μm 6 Aluminum metal layers
  - M1-5 0.08 Ω/square (0.5 μm x 1mm wire = 160 Ω)
  - M6 0.03 Ω/square (0.5 μm x 1mm wire = 60 Ω)

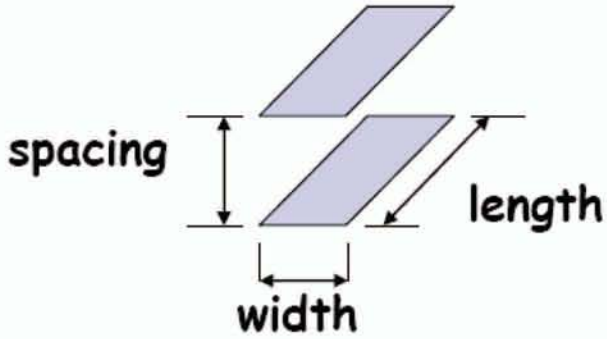
Wire capacitance



- Capacitance depends on geometry of surrounding wires and relative permittivity, ε_r, of insulating dielectric
  - silicon dioxide, SiO_2 $\varepsilon_r = 3.9$
  - silicon flouride, SiOF $\varepsilon_r = 3.1$
  - SiLK™ polymer, $\varepsilon_r = 2.6$
- Can have different materials between wires and between layers, and also different materials on higher layers

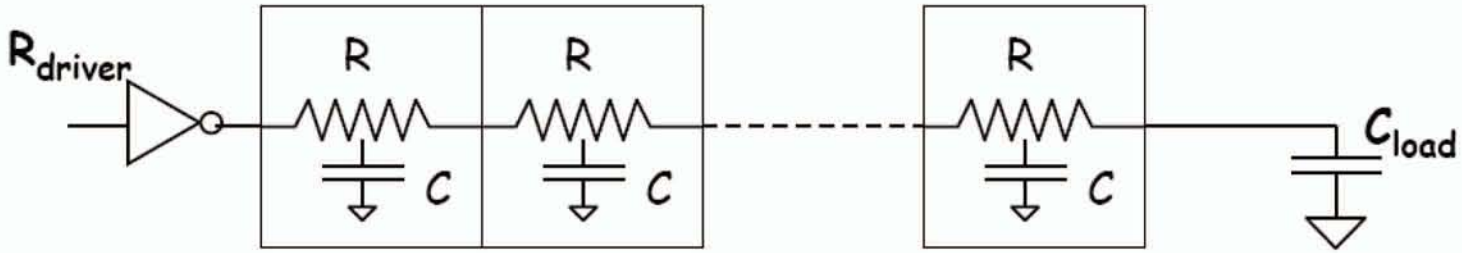
Capacitance scaling

$$\text{parallel plate capacitance} \propto \frac{\text{width}}{\text{spacing}} \times \text{length}$$

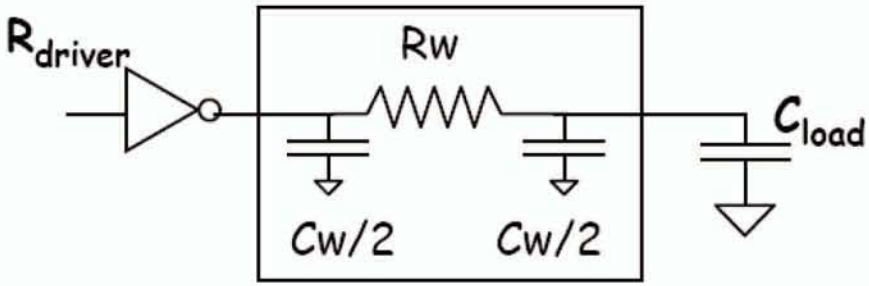


- Capacitance/unit length ~constant with feature size scaling (width and spacing scale together)
  - Isolated wire sees approx. 100 fF/mm
  - With close neighbors about 160 fF/mm
- Need to use capacitance extractor to get accurate values

Wire delay models



- Wire has distributed R and C per unit length
  - wire delay increases quadratically with length
  - edge rate also degrades quadratically with length
- Simple lumped Π model gives reasonable approximation
  - Rw is lumped resistance of wire
  - Cw is lumped capacitance (put half at each end)



$$\text{Delay} = \text{Rdriver} \times \frac{\text{Cw}}{2} + (\text{Rdriver} + \text{Rw}) \times \left(\frac{\text{Cw}}{2} + \text{Cload} \right)$$

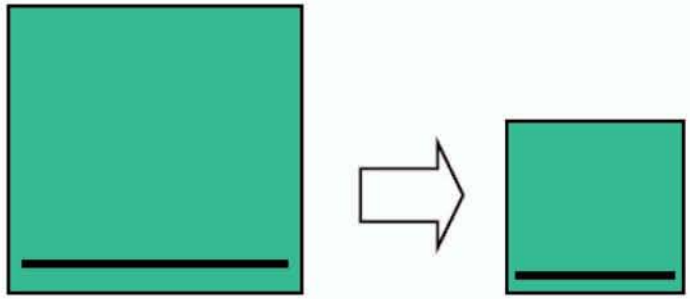
Wire delay example – our technology

- In 0.18μm TSMC, 5x minimum inverter with effective resistance of 3 kΩ, driving FO4 load (25fF)
- Delay = Rdriver x Cload = 75 ps

- Now add 1mm M1 wire, 0.25μm wide
  - Rw = 320Ω wire + 22Ω vias = 344Ω
  - Cw = 160fF

$$\begin{aligned} \text{Delay} &= \text{Rdriver} \times \frac{Cw}{2} + (\text{Rdriver} + \text{Rw}) \times \left(\frac{Cw}{2} + \text{Cload} \right) \\ &= 3\text{k}\Omega \times 80\text{fF} + (3\text{k}\Omega + 344\Omega) \times (80\text{fF} + 25\text{fF}) \\ &= 591\text{ps} \end{aligned}$$

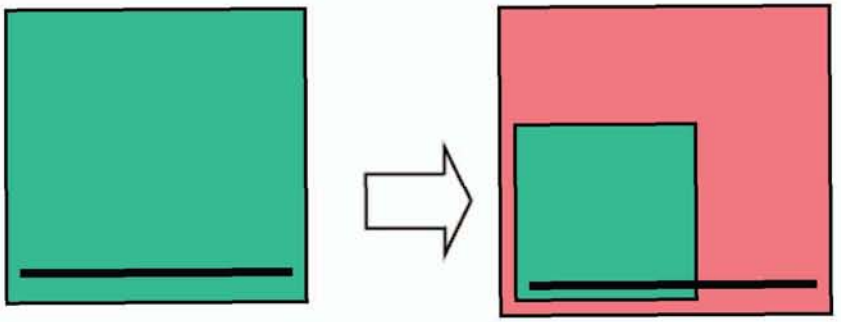
Wire delay scaling, local wires



- For wire crossing same amount of circuitry
  - Resistance stays roughly constant
    - length decreases by same amount as width, height stays large and/or change material to copper
  - Capacitance decreases by scaling factor
    - cap/unit length constant, length decreases
- Wire delay tracks improvement in gate delay

[From Mark Horowitz, DAC 2000]

Wire delay scaling, global wires



- For wire crossing whole chip
  - Resistance grows linearly
  - Capacitance stays fixed
- Wire delay increases relative to gate delay

[From Mark Horowitz, DAC 2000]

Fewer gates per clock cycle

- Processors in Intel 386 generation, around 50 FO4 gate delays per clock cycle
- Pentium-4 around 16 FO4 in normal clock, around 8 FO4 delays in fast ALU section
- Fastest 64-bit adder around 7 FO4 delays
- As measured in distance per clock cycle, wires are getting much slower

Chip area traversed in one clock cycle