

## Administrative matter

- Makeup lecture tonight
  - 6:10PM-7:30PM
    - In Surge 173
  - No laboratory
- Final examination
  - will cover
    - Chapter 8, 10, 11
    - Reliability
    - Scheduling
    - Additional topic from RT Burns & Willings
    - Labs...
  - Weight slightly higher than midterm
    - I will take “improvement” into consideration



## Technology mapping

- Library of gates available for implementation
  - Simple
    - only 2-input AND,OR gates
  - Complex
    - various-input AND,OR,NAND,NOR,etc. gates
    - Efficiently implemented meta-gates (ie., AND-OR-INVERT,MUX)
- Final structure consists of specified library's components only
- If technology mapping integrated with logic synthesis
  - More efficient circuit
  - More complex problem
  - Heuristics required, may result in sub-optimal solution
- Crucial for “pre-fabricated” technologies
  - Gate array, standard cell, FPGA



## Complexity impact on user

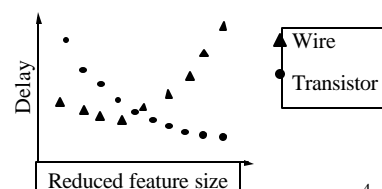
- As complexity grows, heuristics used
- Heuristics differ tremendously among synthesis tools
  - Computationally expensive (slow heuristics)
    - Higher quality results
    - Long run times (hours, days)
    - Requires huge amounts of memory
    - Typically needs to run on servers, workstations
  - Computationally cheap (fast heuristics)
    - Lower quality results
    - Shorter run times (minutes, hours)
    - Smaller amount of memory required
    - Could run on PC
- Super-linear-time (ie.  $n^3$ ) heuristics usually used
  - User can partition large systems to reduce run times/size
  - $100^3 > 50^3 + 50^3$  ( $1,000,000 > 250,000$ )
    - But the result may be suboptimal: can't optimize across boundary

3

CS122A: Embedded System Design, Fall 01

## Integrating logic design and physical design

- Past
  - Gate delay much greater than wire delay
  - Thus, performance evaluated as # of levels of gates only
- Today
  - Gate delay shrinking as feature size shrinking
  - Wire delay increasing
    - Performance evaluation needs wire length
  - Transistor placement impact wire length
    - Integrating with physical design
    - Wire-planning (simultaneous routing and logic synthesis)
  - Simultaneous logic synthesis and physical design required



4

CS122A: Embedded System Design, Fall 01

## Register-transfer synthesis

- Converts FSMD to custom single-purpose processor
  - Datapath
    - Register units to store variables (Complex data types)
    - Functional units (Arithmetic operations)
    - Connection units (Busses, MUXs)
  - FSM controller
    - Controls datapath
- Key subproblems:
  - Allocation
    - Instantiate storage, functional, connection units
  - Binding
    - Mapping FSMD operations to specific units



CS122A: Embedded System Design, Fall 01

5

## Behavioral synthesis

- High-level synthesis
- Converts single sequential program to GPP and SPP
- Key subproblems
  - Allocation
  - Binding
  - Scheduling
    - Assign sequential program's operations to states
    - Conversion template for SPP (ESD chapter 3)
- Optimizations important
  - Compiler
    - Constant propagation, dead-code elimination, loop unrolling
  - Advanced techniques for allocation, binding, scheduling



CS122A: Embedded System Design, Fall 01

6

## System synthesis

---

- Convert 1 or more processes onto 1 or more processors (system)
  - For complex embedded systems
    - Multiple processes may provide better performance/power
    - May be better described using concurrent sequential programs
- Tasks
  - Transformation
    - Can merge 2 exclusive processes into 1 process
    - Can break 1 large process into separate processes
    - Procedure inlining
    - Loop unrolling
  - Allocation
    - Design of system architecture
    - Select processors to implement processes
    - Also select memories and busses



CS122A: Embedded System Design, Fall 01

7

## System synthesis

---

- Tasks (cont.)
  - Partitioning
    - Mapping 1 or more processes to 1 or more processors
    - Mapping variables to memories
    - Mapping communications to buses
  - Scheduling
    - Multiple processes on a single processor
    - Memory accesses
    - Bus communications
- Synthesis driven by constraints
  - E.g., Meet performance requirements at minimum cost
    - Allocate as much behavior as possible to general-purpose processor
    - Low-cost/flexible implementation
    - Minimum # of SPPs used to meet performance



CS122A: Embedded System Design, Fall 01

8

## System synthesis

- System synthesis for GPP only (software)
  - Common for decades
    - Multiprocessing
    - Parallel processing
    - Real-time scheduling
- Hardware/software codesign
  - Simultaneous consideration of GPPs/SPPs during synthesis
  - Made possible by maturation of behavioral synthesis in 1990's



CS122A: Embedded System Design, Fall 01

9

## Temporal vs. spatial thinking

- Design thought process changed by evolution of synthesis
- Before synthesis tools come along
  - Designers worked primarily in structural domain
    - Connecting simpler components to build more complex systems
    - Connecting logic gates to build controller
    - Connecting registers, MUXs, ALUs to build datapath
  - “capture and simulate” era
    - Capture using CAD tools
    - Simulate to verify correctness before fabricating
  - Spatial thinking
    - Structural diagrams
    - Data sheets
  - Bottom-up methodologies



CS122A: Embedded System Design, Fall 01

10

## Temporal vs. spatial thinking

---

- After synthesis tools come along
  - Designers work primarily in behavioral domain
  - “describe and synthesize” era
    - Describe FSMs or sequential programs
    - Synthesize into structure
  - Top-down methodologies
  - Temporal thinking
    - States or sequential statements have relationship over time
- Strong understanding of hardware structure still important
  - Behavioral description must synthesize to efficient structural implementation



CS122A: Embedded System Design, Fall 01

11

## Outline

---

- Automation: synthesis
- Verification: hardware/software co-simulation
- Reuse: intellectual property cores
- Design process models



CS122A: Embedded System Design, Fall 01

12

## Verification

---

- Ensuring design is correct and complete
  - Correct
    - Implements specification accurately
  - Complete
    - Describes appropriate output to all relevant input
- Formal verification
  - Computationally difficult
    - HOL (Theorem Prover), SMV (Model Checker)
  - For small designs or verifying certain key properties only
- Simulation
  - Most common verification method



## Formal verification

---

- Analyze design to prove or disprove certain properties
- Correctness example
  - Prove ALU structural implementation equivalent to behavioral description
    - Derive Boolean equations for outputs (non-canonical)
    - Create truth table for equations
    - Compare to truth table from original behavior
- Completeness example
  - Formally prove elevator door can never open while elevator is moving
    - Derive conditions for door being open
    - Show conditions conflict with conditions for elevator moving



## Simulation

- Create computer model of design
  - Provide sample input
  - Check for acceptable output
- Correctness example
  - ALU
    - Provide all possible input combinations
    - Check outputs for correct results
- Completeness example
  - Elevator door closed when moving
    - Provide all possible input sequences
    - Check door always closed when elevator moving

15

*CS122A: Embedded System Design, Fall 01*

## Increases confidence

- Simulating all possible input sequences impossible for most systems
  - E.g., 32-bit ALU
    - $2^{32} * 2^{32} = 2^{64}$  possible input combinations
    - At 1 million combinations/sec
    - ½ million years to simulate
    - Sequential circuits even worse
- Can only simulate tiny subset of possible inputs
  - Typical values
  - Known boundary conditions
    - E.g., 32-bit ALU
    - Both operands all 0's
    - Both operands all 1's
- Increases confidence of correctness/completeness
- Does not prove

16

*CS122A: Embedded System Design, Fall 01*

## Advantages over physical implementation

---

- **Controllability**
  - Control time
    - Stop/start simulation at any time
  - Control data values
    - Inputs or internal values
- **Observability**
  - Examine system/environment values at any time
- **Debugging**
  - Can stop simulation at any point and:
    - Observe internal values
    - Modify system/environment values before restarting
  - Can step through small intervals (i.e., 500 nanoseconds)



CS122A: Embedded System Design, Fall 01

17

## Disadvantages

---

- **Simulation setup time**
  - Often has complex external environments
  - Could spend more time modeling environment than system
- **Models likely incomplete**
  - Some environment behavior may be undocumented
  - May not model behavior correctly
- **Simulation speed much slower than actual execution**
  - Sequentializing parallel design
    - IC: gates operate in parallel
    - Simulation: analyze inputs, generate outputs for each gate 1 at time
  - Several programs added between simulated system and real hardware
    - 1 simulated operation:
    - = 10 to 100 simulator operations

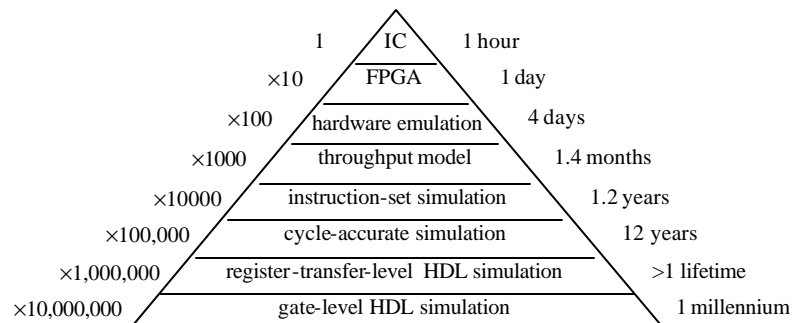


CS122A: Embedded System Design, Fall 01

18

## Simulation speed

- Relative speeds of different types of simulation/emulation
  - 1 hour actual execution of SOC
    - = 1.2 years instruction-set simulation
    - = 10,000,000 hours gate-level simulation



19

CS122A: Embedded System Design, Fall 01

## Overcoming long simulation time

- Reduce amount of real time simulated
  - 1 msec execution instead of 1 hour
    - $0.001\text{sec} * 10,000,000 = 10,000\text{ sec} = 3\text{ hours}$
  - Reduced confidence
- Faster simulator
  - Emulators
    - Special hardware for simulations
    - Exchange speed for observability/controllability
  - Less precise/accurate simulators

20

CS122A: Embedded System Design, Fall 01

## Reducing precision/accuracy

- Don't need gate-level analysis for all simulations
  - E.g., cruise control
    - Don't care what happens at every input/output of each logic gate
  - Simulating RT components ~10x faster
  - Cycle-based simulation ~100x faster
    - Accurate at clock boundaries only
    - No information on signal changes between boundaries
- Faster simulator often combined with reduction in real time
  - If willing to simulate for 10 hours
    - Use instruction-set simulator which is 10,000 times slower
    - Real execution time simulated
    - 10 hours \* 1 / 10,000
    - = 0.001 hour
    - = 3.6 seconds



CS122A: Embedded System Design, Fall 01

21

## Hardware/software co-simulation

- Variety of simulation approaches exist
  - From very detailed
    - E.g., gate-level model
  - To very abstract
    - E.g., instruction-level model
- Simulation tools evolved separately for hardware/software
  - Recall separate design evolution
  - Software (GPP)
    - Typically with instruction-set simulator (ISS)
  - Hardware (SPP)
    - Typically with models in HDL environment
- Integration of GPP/SPP on single IC creating need for merging simulation tools



CS122A: Embedded System Design, Fall 01

22

## Integrating GPP/SPP simulations

- **Simple/naïve way**
  - HDL model of microprocessor
    - Runs system software
    - Much slower than ISS
    - Less observable/controllable than ISS
  - HDL models of SPPs
  - Integrate all models
- **Hardware-software co-simulator**
  - ISS for microprocessor
  - HDL model for SPPs
  - Create communication between simulators
  - Simulators run separately except when transferring data
  - Faster
    - Frequent communication between ISS and HDL model slows it down



CS122A: Embedded System Design, Fall 01

23

## Minimizing communication

- **Memory shared between GPP and SPPs**
  - Where should memory go?
    - In ISS
      - HDL simulator must stall for memory access
    - In HDL?
      - ISS must stall when fetching each instruction
- **Model memory in both ISS and HDL**
  - Most accesses by each model unrelated to other's accesses
    - No need to communicate these between models
  - Co-simulator ensures consistency of shared data



CS122A: Embedded System Design, Fall 01

24

## Emulators

- General physical device system mapped to
  - Microprocessor emulator
    - Microprocessor IC with some monitoring, control circuitry
  - SPP emulator
    - FPGAs (10s to 100s)
  - Usually supports debugging tasks
- Created to help solve simulation disadvantages
  - Mapped relatively quickly
    - Hours, days
  - Can be placed in real environment
    - No environment setup time
    - No incomplete environment
  - Typically faster than simulation
    - Hardware implementation

25

*CS122A: Embedded System Design, Fall 01*

## Disadvantages

- Still not as fast as real implementations
  - E.g., emulated engine controller may not respond fast enough to keep control of engine
- Mapping still time consuming
  - E.g., mapping complex SOC to 10 FPGAs
    - Just partitioning into 10 parts could take a long time
- Can be very expensive
  - Top-of-the-line FPGA-based emulator: \$100,000 to \$1mill
    - Cadence/Quickturn
  - Leads to resource bottleneck
    - Can maybe only afford 1 emulator
    - Groups wait days, weeks for other group to finish using it

26

*CS122A: Embedded System Design, Fall 01*

## Outline

---

- Automation: synthesis
- Verification: hardware/software co-simulation
- Reuse: intellectual property cores
- Design process models



CS122A: Embedded System Design, Fall 01

27

## Reuse: intellectual property cores

---

- Commercial off-the-shelf (COTS) components
  - Predesigned, prepackaged ICs
  - Implements GPP or SPP
  - Reduces design/debug time
  - Have always been available
- System-on-a-chip (SOC)
  - All components of system implemented on single chip
  - Made possible by increasing IC capacities
  - Changing the way COTS components sold
    - As intellectual property (IP) rather than actual IC
    - Behavioral, structural, or physical descriptions
    - Processor-level components known as cores
    - SOC built by integrating multiple descriptions



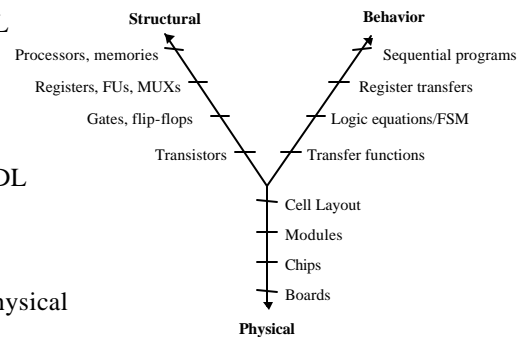
CS122A: Embedded System Design, Fall 01

28

## Cores

- **Soft core**
  - Synthesizable behavioral description
  - Typically written in HDL (VHDL/Verilog)
- **Firm core**
  - Structural description
  - Typically provided in HDL
- **Hard core**
  - Physical description
  - Provided in variety of physical layout file formats

**Gajski's Y-chart**



29

CS122A: Embedded System Design, Fall 01

## Advantages/disadvantages of hard core

- **Ease of use**
  - Developer already designed and tested core
    - Can be used right away
    - Can expect to work correctly
- **Predictability**
  - Size, power, performance predicted accurately
- **Not easily mapped (retargeted) to different process**
  - E.g., core available for vendor X's 0.25 um CMOS process
    - Can't use with vendor X's 0.18 um process
    - Can't use with vendor Y

30

CS122A: Embedded System Design, Fall 01

## Advantages/disadvantages of soft/firm cores

- **Soft cores**
  - Can be synthesized to nearly any technology
  - Can optimize for particular use
    - E.g., delete unused portion of core for lower power, smaller designs
  - Requires more design effort
  - May not work in technology not tested for
  - Not as optimized as hard core for same processor
- **Firm cores**
  - Compromise between hard and soft cores
    - Some retargetability
    - Limited optimization
    - Better predictability/ease of use

31

*CS122A: Embedded System Design, Fall 01*

## New challenges to processor providers

- **Cores have dramatically changed business model**
- **Pricing models**
  - Past
    - Vendors sold product as IC to designers
    - Designers must buy any additional copies
    - Could not (economically) copy from original
  - Today
    - Vendors can sell as IP
    - Designers can make as many copies as needed
- **Vendor can use different pricing models**
  - Royalty-based model
    - Similar to old IC model
    - Designer pays for each additional model
  - Fixed price model
    - One price for IP and as many copies as needed
  - Many other models used

32

*CS122A: Embedded System Design, Fall 01*

## IP protection

---

- **Past**
  - Illegally copying IC very difficult
    - Reverse engineering required tremendous, deliberate effort
    - “Accidental” copying not possible
- **Today**
  - Cores sold in electronic format
    - Deliberate/accidental unauthorized copying easier
    - Contracts to ensure no copying/distributing
  - Encryption techniques
    - limit actual exposure to IP
  - Watermarking
    - determines if particular instance of processor was copied
    - whether copy authorized



CS122A: Embedded System Design, Fall 01

33

## New challenges to processor users

---

- **Licensing arrangements**
  - Not as easy as purchasing IC
  - More contracts enforcing pricing model and IP protection
    - Possibly requiring legal assistance
- **Extra design effort**
  - Especially for soft cores
    - Must still be synthesized and tested
    - Minor differences in synthesis tools can cause problems
- **Verification requirements more difficult**
  - Extensive testing for synthesized soft/firm cores mapped to particular technology
    - Ensure correct synthesis
    - Timing and power vary between implementations
  - Early verification critical
    - Cores buried within IC
    - Cannot simply replace bad core



CS122A: Embedded System Design, Fall 01

34

## Outline

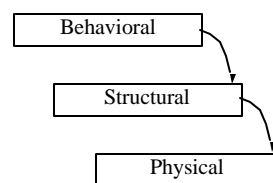
- Automation: synthesis
- Verification: hardware/software co-simulation
- Reuse: intellectual property cores
- Design process models
  - Methodologies



## Design process model

- Order that design steps are processed
  - Behavior description step
  - Behavior to structure conversion step
  - Mapping structure to physical implementation

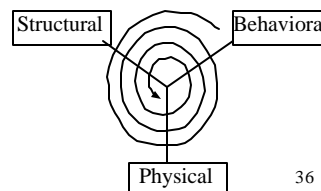
### Waterfall design model



- Waterfall model
  - Go to next only after current step completed

- Spiral model
  - Proceed through steps in order but less detail
  - Repeat steps gradually increasing detail
  - Repeating until desired system obtained

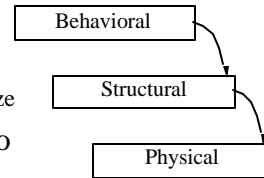
### Spiral design model



## Waterfall method

- Not very realistic
  - Bugs often found in later steps that must be fixed in earlier step
    - E.g., forgot to handle certain input condition
  - Prototype often needed to know complete desired behavior
    - E.g., customer adds features after product demo
  - System specifications commonly change
    - E.g., remain competitive by reducing power, size
- Unexpected iterations may increase time to market
  - Lost revenues

**Waterfall design model**



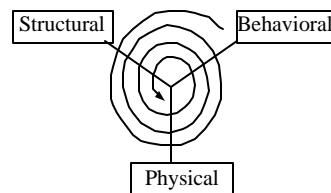
37

CS122A: Embedded System Design, Fall 01

## Spiral method

- Iteration of steps may be incomplete
- Much faster, though
  - End up with prototype at a particular level
    - Use to test basic functions
    - Get idea of functions to add/remove
  - Interactivity
    - Experience applies to later iterations
- Quick implementation needed
  - E.g., FPGAs for prototype
    - silicon for final product
  - May have to use more tools
    - Extra effort/cost
- May require more time
  - Can't really have right-first-time

**Spiral design model**

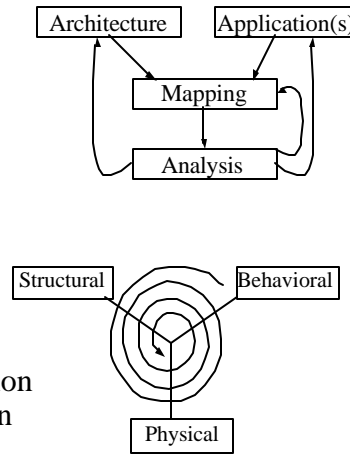


38

CS122A: Embedded System Design, Fall 01

## Spiral model

- Designer develops or acquires architecture
- Develops application(s)
- Maps application to architecture
- Analyzes design metrics
- Now makes choice
  - Modify mapping
  - Modify application(s) to better suit architecture
  - Modify architecture to better suit application(s)
- Continue refining to lower abstraction level until particular implementation chosen



39

CS122A: Embedded System Design, Fall 01

## IC Technology

40

## Outline

- Anatomy of integrated circuits
- Full-Custom (VLSI) IC Technology
- Semi-Custom (ASIC) IC Technology
- Programmable Logic Device (PLD) IC Technology

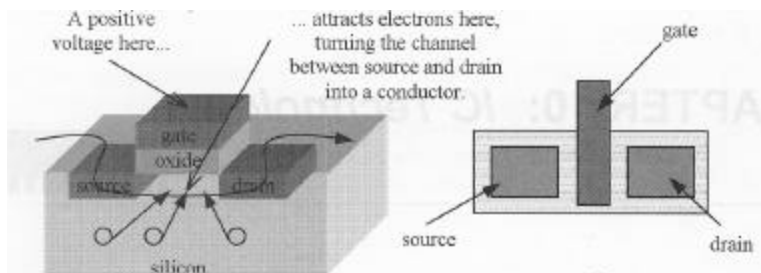


CS122A: Embedded System Design, Fall 01

41

## CMOS transistor

- Source, Drain
  - Diffusion area where electrons can flow
  - Can be connected to metal contacts (via's)
- Gate
  - Polysilicon area where control voltage is applied
- Oxide
  - Si O<sub>2</sub> Insulator so the gate voltage can't leak

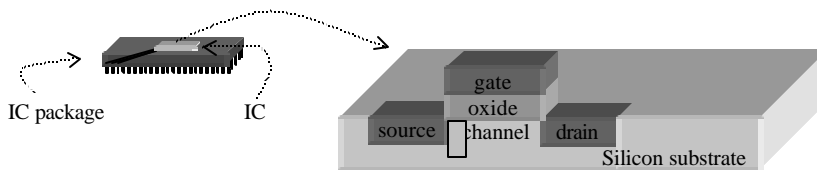


CS122A: Embedded System Design, Fall 01

42

## End of the Moore's Law?

- Every dimension of the MOSFET has to scale
  - (PMOS) Gate oxide has to scale down to
    - Increase gate capacitance
    - Reduce leakage current from S to D
    - Pinch off current from source to drain
  - Current gate oxide thickness is about 2.5-3nm
- That's about 25 atoms!!!

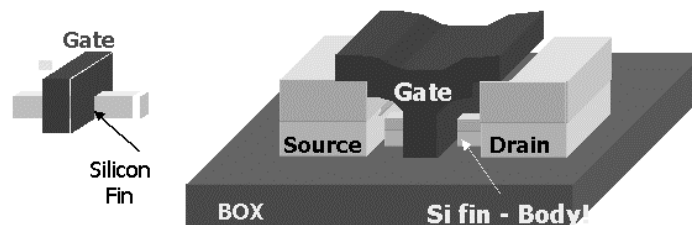


43

CS122A: Embedded System Design, Fall 01

## Proposed Structures: FinFET

Body is a Thin Silicon Film  
Double Gate Structure + Raised Source Drain



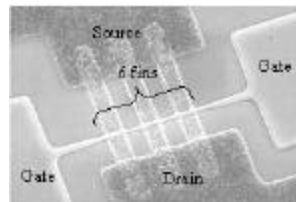
X. Huang, et al, 1999 IEDM, p.67-70

44

CS122A: Embedded System Design, Fall 01

## 20Ghz +

- FinFET has been manufactured to 18nm
  - Still acts as a very good transistor
  - Today, industry leader at 130nm
- Simulation shown that it can be scaled to 10nm
  - Quantum effect start to kick in
    - Reduce mobility by ~10%
  - Ballistic transport become significant
    - Increase current by about ~20%

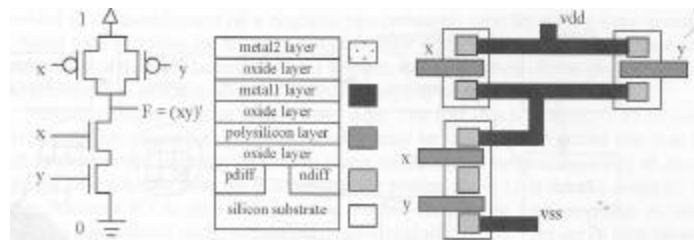


45

CS122A: Embedded System Design, Fall 01

## NAND

- Metal layers for routing (~10)
- PMOS don't like 0
- NMOS don't like 1
- A stick diagram form the basis for mask sets

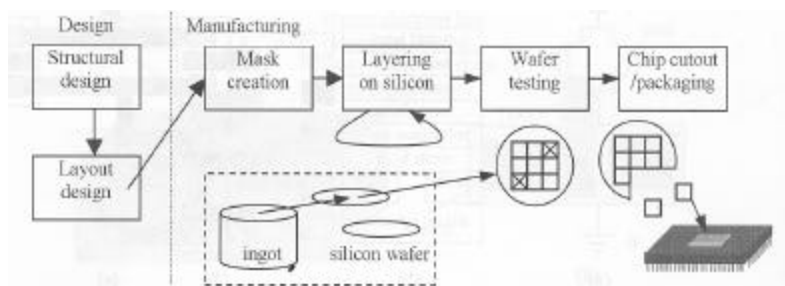


46

CS122A: Embedded System Design, Fall 01

## Silicon manufacturing steps

- Tape out
  - Send design to manufacturing
- Spin
  - One time through the manufacturing process
- Photolithography
  - Drawing patterns by using photoresist to form barriers for deposition

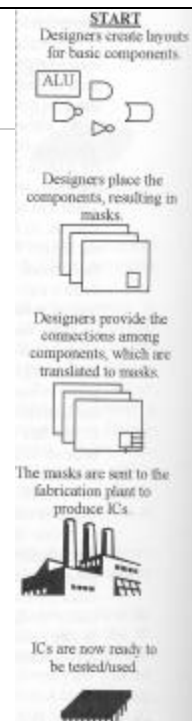


47

CS122A: Embedded System Design, Fall 01

## Full Custom

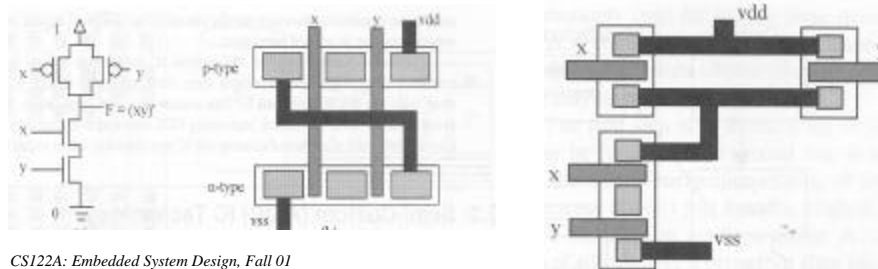
- Very Large Scale Integration (VLSI)
- Placement
  - Place and orient transistors
- Routing
  - Connect transistors
- Sizing
  - Make fat, fast wires or thin, slow wires
  - May also need to size buffer
- Design Rules
  - “simple” rules for correct circuit function
    - Metal/metal spacing, min poly width...



CS122A: Embedded System Design, Fall 01

## Full Custom

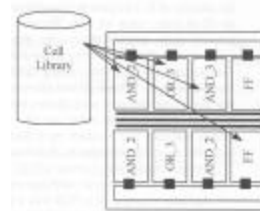
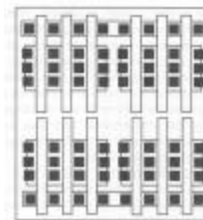
- Best size, power, performance
- Hand design
  - Horrible time-to-market/flexibility/NRE cost...
  - Reserve for the most important units in a processor
    - ALU, Instruction fetch...
- Physical design tools
  - Less optimal, but faster...



CS122A: Embedded System Design, Fall 01

## Semi-Custom

- Gate Array
  - Array of prefabricated gates
  - “place” and route
  - Higher density, faster time-to-market
  - Does not integrate as well with full-custom
- Standard Cell
  - A library of pre-designed cell
  - Place and route
  - Lower density, higher complexity
  - Integrate great with full-custom



50

CS122A: Embedded System Design, Fall 01

## Semi-Custom

- Most popular design style
- Jack of all trade
  - Good
    - Power, time-to-market, performance, NRE cost, per-unit cost, area...
- Master of none
  - Integrate with full custom for critical regions of design

**Gate array**

Designers are provided with a set of masks of predefined gates.

**START**  
Designers provide the connections among gates, which are translated to masks.

The masks are sent to the fabrication plant to produce ICs.

ICs are now ready to be tested/used.

**Standard cell**

Designers are provided with a library of predefined cells.

**START**  
Designers choose cells, place and connect them, resulting in masks.

Designers provide the connections among cells, which are translated to masks.

The masks are sent to the fabrication plant to produce ICs.

ICs are now ready to be tested/used.

CS122A: Embedded System Design, Fall 01

### Programmable Logic Array (PLA)

*PLA exploits structure of expression.*

$F = AB + C$

$G = AB + \bar{C} + \bar{A}\bar{B}$

A
 $\bar{A}$ 
B
 $\bar{B}$ 
C
 $\bar{C}$ 
F
G

13

## Programmable Logic Device

- Programmable Logic Device
  - Programmable Logic Array, Programmable Array Logic, Field Programmable Gate Array
- All layers already exist
  - Designers can purchase an IC
  - To implement desired functionality
    - Connections on the IC are either created or destroyed to implement
- Benefits
  - Very low NRE costs
  - Great time to market
- Drawback
  - High unit cost, bad for large volume
  - Power
    - Except special PLA
  - slower



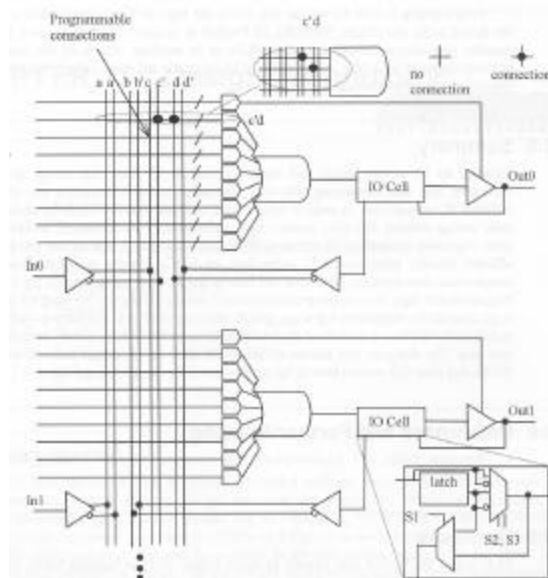
1600 usable gate, 7.5 ns  
\$7 list price

53

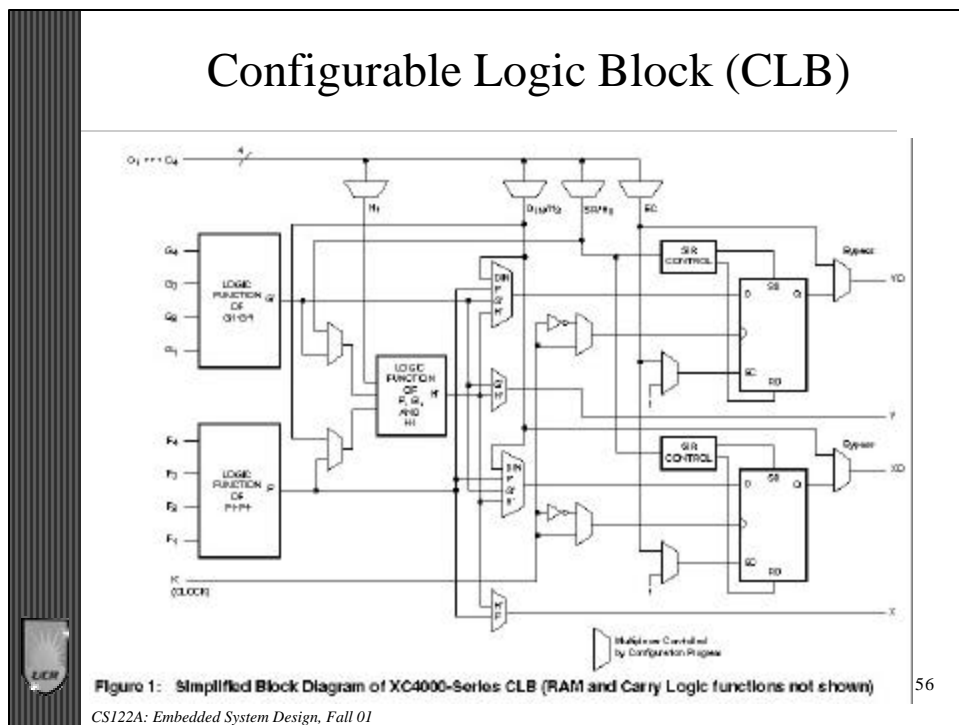
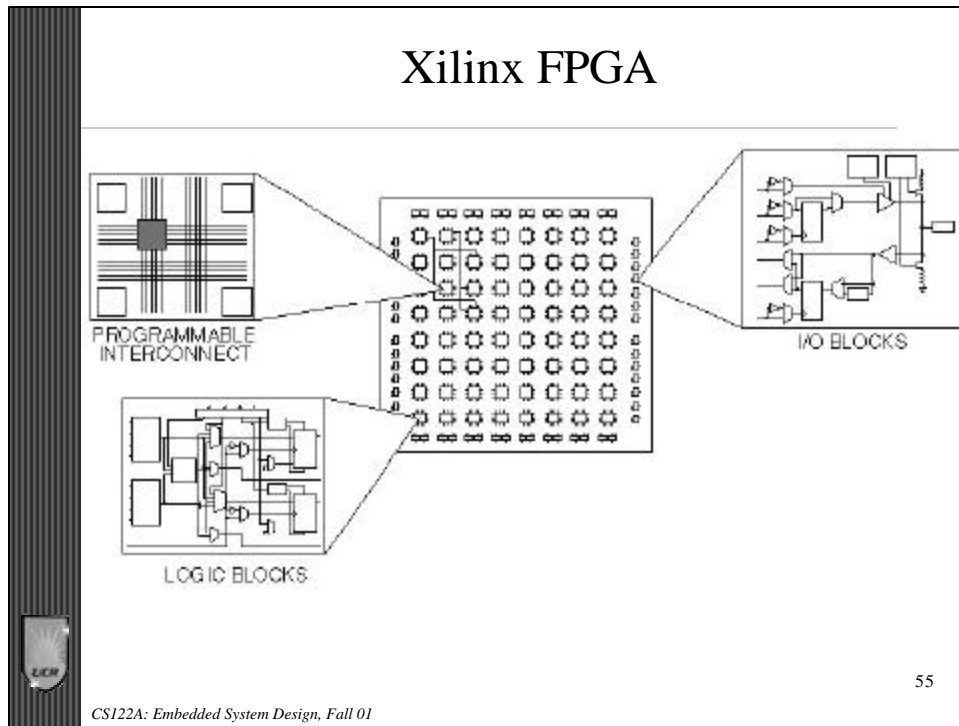
CS122A: Embedded System Design, Fall 01

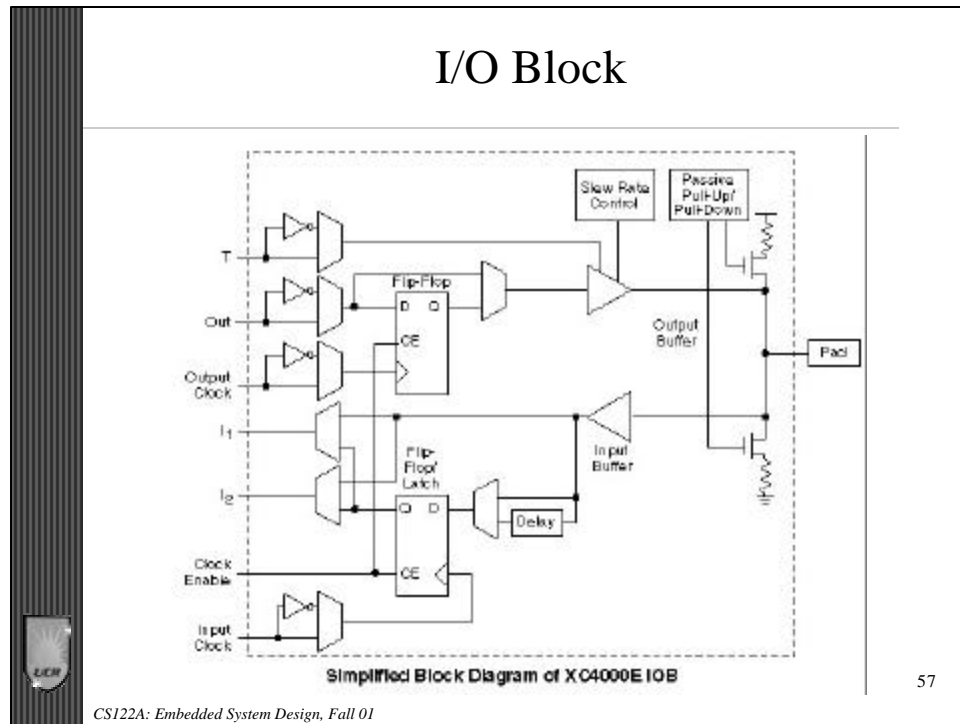
## Complex PLA

- Adding I/O function
- Multiple PLA on the same chip
- Connection between PLAs
- Does not scale well



CS122A: Embedded System Design, Fall 01





- IC Technology

