ESE5320:
System-on-a-Chip Architecture

Day 1: August 30, 2023

Introduction and Overview

(lecture start target 10:20am)

Note: slides linked to web
www.seas.upenn.edu/~ese5320/fall2023/fall2023.html

- Preclass (work now)
- Feedback form (turn in end of lecture)

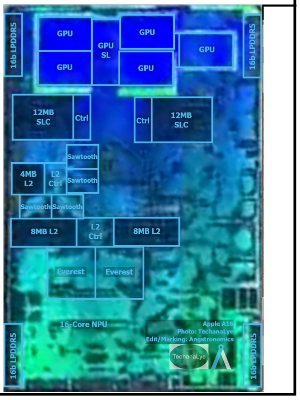
Penn

Today

- Part 1: Case for Programmable SoC (motivation)
- Part 2: Course Goals, Outcomes, Tools (philosophy?)
- Part 3: Sample Optimization (fast, flavor)
- Part 4: This course (operational details)
  - (including policies, logistics)

Apple A16 Bionic

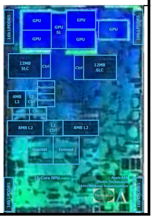
- ? 110+mm^2, 4nm
- 16 Billion Tr.
- iPhone 14
- 6 ARM cores
  - 2 fast (3.5GHz)
  - 4 low energy (2GHz)
- 5 custom GPUs (1.4GHz)
- 16 Neural Engines
  - 17 Trillion ops/s?



https://en.wikipedia.org/wiki/Apple_A16

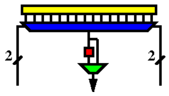
Questions

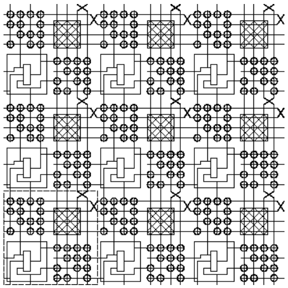
- Why do today's SoC look like they do?
- How approach programming modern SoCs?
- How design a custom SoC?
- When building a System-on-a-Chip (SoC)
  - How much area should go into:
    - Processor cores, GPUs, FPGA logic, memory, interconnect, custom functions (which) …. ?



FPGA
Field-Programmable Gate Array

K-LUT (typical k=4 or 6)
Compute block
w/ optional
output Flip-Flop





ESE1500, CIS5710

Case for Programmable SoC

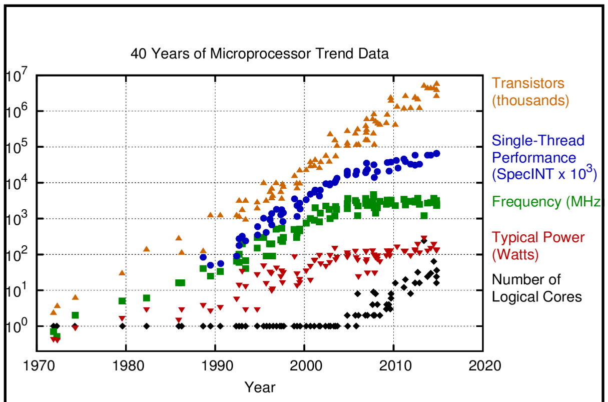
End of microprocessor Scaling

Old

- Moore's Law scaling delivered faster transistors
- Processors rode Moore's Law
  - Turning transistors into performance
- Could wait and ride technology curve

Now

- Dennard's Law kicked in
  - How need to scale voltage with size
- microprocessors were burning more power
- Lost ability to scale down voltage
- Processor performance stalled



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

The Way things Were

30 years ago

- Wanted programmability
  - used a processor
- Wanted it a little faster
  - Next year's processor would run faster…
- Wanted high-throughput
  - used a custom Integrated Circuit (IC) -- chip
- Wanted product differentiation
  - Got it at the board level
  - Select which ICs and how wired
- Build a custom IC (chip)
  - It was about gates and logic

Today

- Microprocessor may not be fast enough
  - (but often it is)
  - Or low enough energy
- Single core processor scaling has ended
- Time and Cost of a custom IC is too high
  - $100M's of dollars for development, Years
- FPGAs promising
  - But build everything from prog. gates?
- Premium for small part count
  - And avoid chip crossing
  - ICs with 10—100 Billions of Transistors

Non-Recurring Engineering (NRE) Costs

- Costs spent up front on development
  - Engineering Design Time
  - Design Verification
  - Prototypes
  - Mask costs
- Recurring Engineering
  - Costs to produce each chip

$$Cost(N_{chips}) = Cost_{NRE} + N_{chips} \times Cost_{perchip}$$

NRE Costs

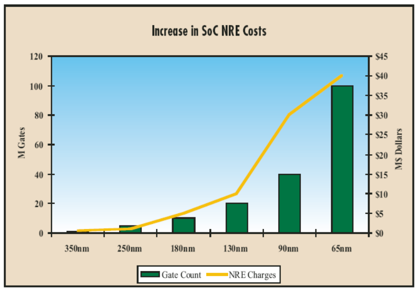
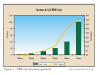
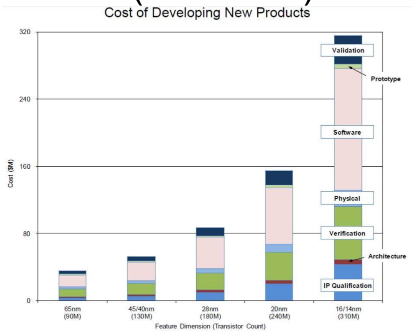


Figure 1 - NRE costs by process geometry

Source: Semico Research Corp.

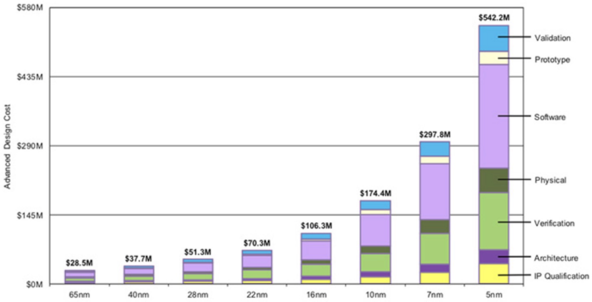
NRE Cost (continued)





https://semiengineering.com/how-much-will-that-chip-cost/

Bonus: Chip Design Costs



https://wccftech.com/apple-5nm-3nm-cost-transistors/

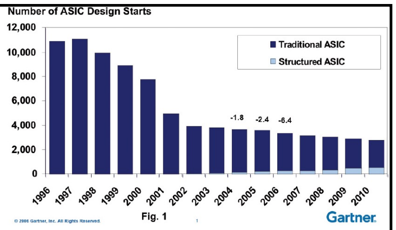
Amortize NRE with Volume

$$Cost(N_{chips}) = Cost_{NRE} + N_{chips} \times Cost_{perchip}$$

$$Cost = \frac{Cost_{NRE}}{N_{chips}} + Cost_{perchip}$$

Economics

Forcing fewer, more customizable chips



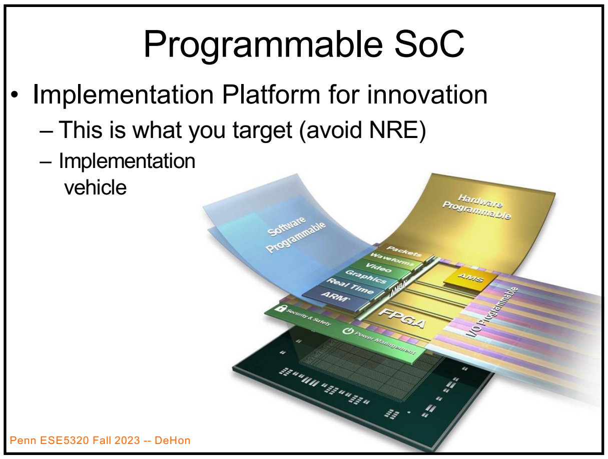
- Economics force fewer, more customizable chips
  - Mask costs in the millions of dollars
  - Custom IC design NRE 100s of millions of dollars
    - Need market of billions of dollars to recoup investment
    - With fixed or slowly growing total IC industry revenues
    - ➔ Number of unique chips must decrease
    - Chips must be programmable

Large ICs (Chips)

- Now contain significant software
  - Almost all have embedded processors
- Must co-design SW and HW
- Must solve complete computing task
  - Tasks has components with variety of needs
  - Some don’t need custom circuit
  - 90/10 Rule

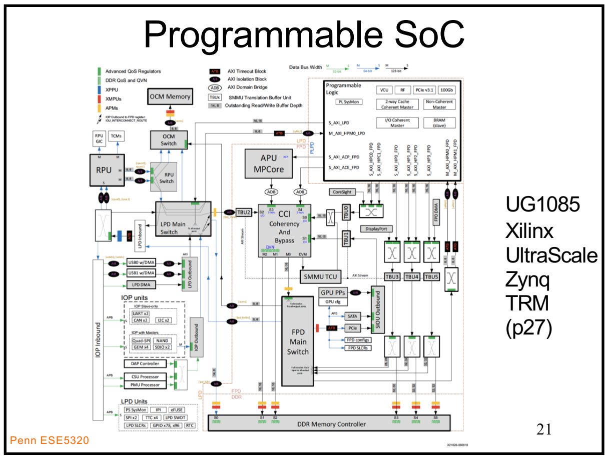
Given Demand for Programmable

- How do we get higher performance than a processor, while retaining programmability?
  - Programmability – don’t have to spend 100s of millions of dollars and months for fabrication?



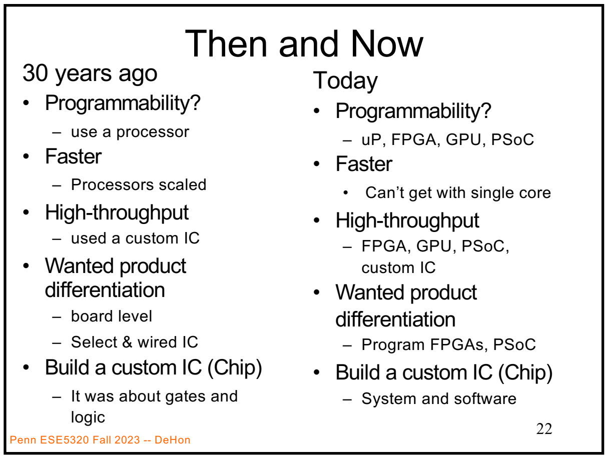
Programmable SoC

- Implementation Platform for innovation
  - This is what you target (avoid NRE)
  - Implementation vehicle



Programmable SoC

UG1085 Xilinx UltraScale Zynq TRM (p27)



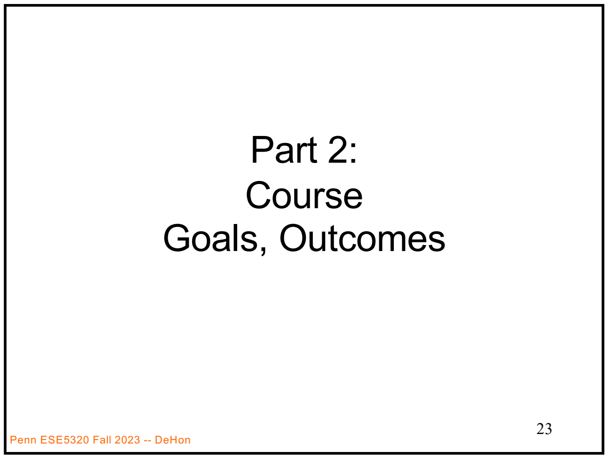
Then and Now

30 years ago

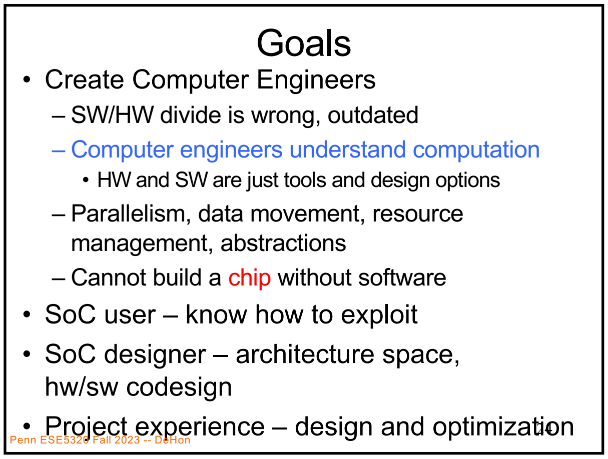
- Programmability?
  - use a processor
- Faster
  - Processors scaled
- High-throughput
  - used a custom IC
- Wanted product differentiation
  - board level
  - Select & wired IC
- Build a custom IC (Chip)
  - It was about gates and logic

Today

- Programmability?
  - uP, FPGA, GPU, PSoC
- Faster
  - Can't get with single core
- High-throughput
  - FPGA, GPU, PSoC, custom IC
- Wanted product differentiation
  - Program FPGAs, PSoC
- Build a custom IC (Chip)
  - System and software

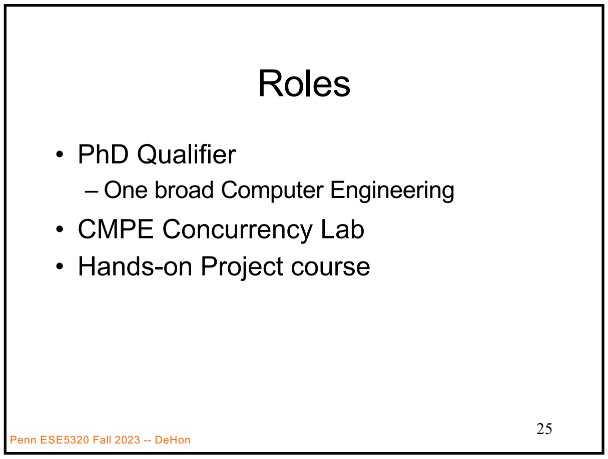


Part 2: Course Goals, Outcomes



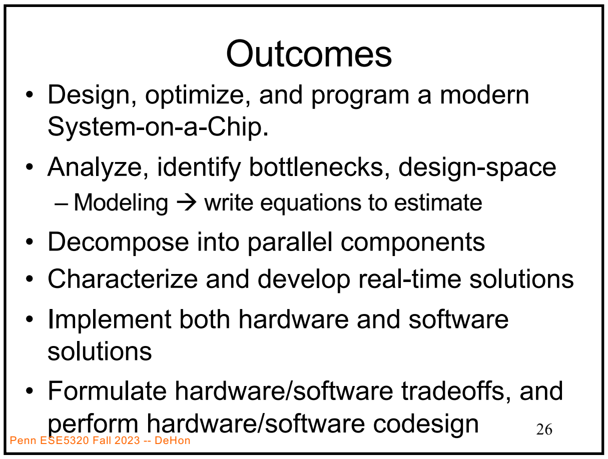
Goals

- Create Computer Engineers
  - SW/HW divide is wrong, outdated
  - Computer engineers understand computation
    - HW and SW are just tools and design options
  - Parallelism, data movement, resource management, abstractions
  - Cannot build a chip without software
- SoC user – know how to exploit
- SoC designer – architecture space, hw/sw codesign
- Project experience – design and optimization



Roles

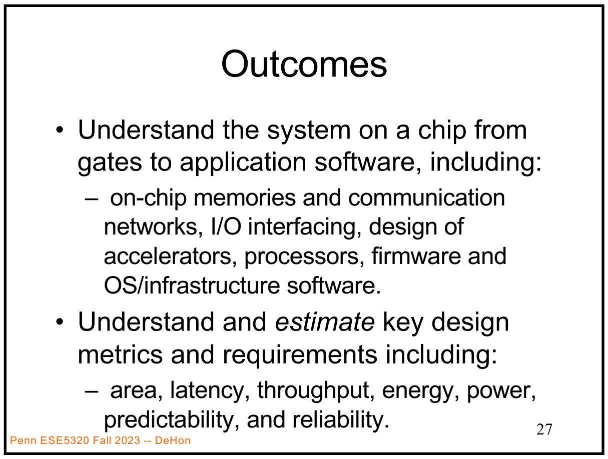
- PhD Qualifier
  - One broad Computer Engineering
- CMPE Concurrency Lab
- Hands-on Project course



Outcomes

- Design, optimize, and program a modern System-on-a-Chip.
- Analyze, identify bottlenecks, design-space
  - Modeling → write equations to estimate
- Decompose into parallel components
- Characterize and develop real-time solutions
- Implement both hardware and software solutions
- Formulate hardware/software tradeoffs, and perform hardware/software codesign

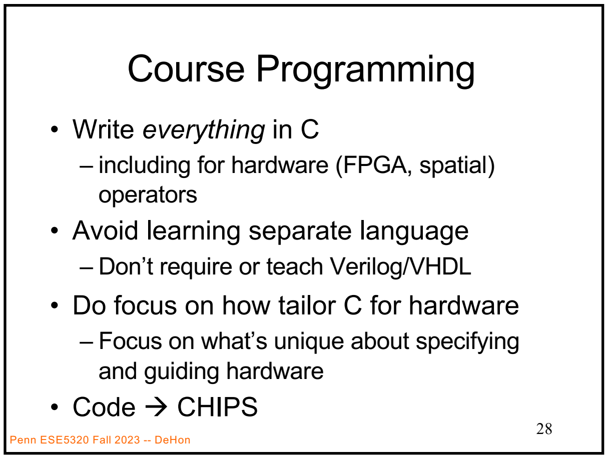
26



Outcomes

- Understand the system on a chip from gates to application software, including:
  - on-chip memories and communication networks, I/O interfacing, design of accelerators, processors, firmware and OS/infrastructure software.
- Understand and *estimate* key design metrics and requirements including:
  - area, latency, throughput, energy, power, predictability, and reliability.

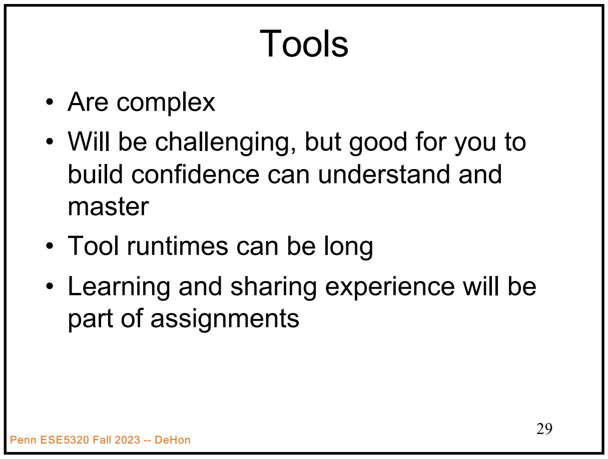
27



Course Programming

- Write *everything* in C
  - including for hardware (FPGA, spatial) operators
- Avoid learning separate language
  - Don't require or teach Verilog/VHDL
- Do focus on how tailor C for hardware
  - Focus on what's unique about specifying and guiding hardware
- Code → CHIPS

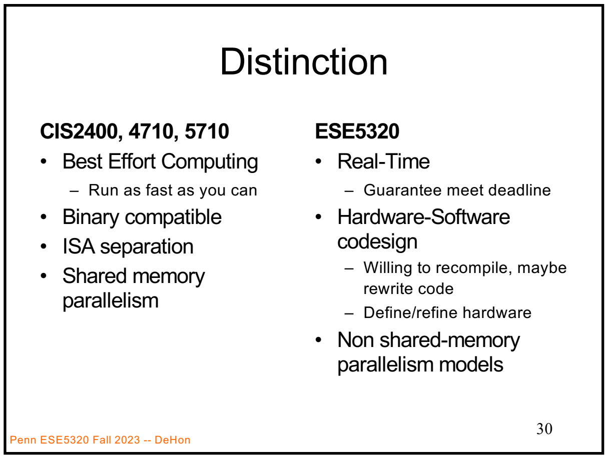
28



Tools

- Are complex
- Will be challenging, but good for you to build confidence can understand and master
- Tool runtimes can be long
- Learning and sharing experience will be part of assignments

29



Distinction

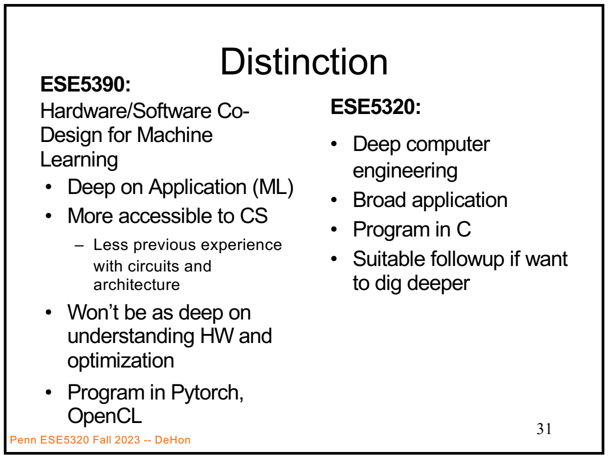
CIS2400, 4710, 5710

- Best Effort Computing
  - Run as fast as you can
- Binary compatible
- ISA separation
- Shared memory parallelism

ESE5320

- Real-Time
  - Guarantee meet deadline
- Hardware-Software codesign
  - Willing to recompile, maybe rewrite code
  - Define/refine hardware
- Non shared-memory parallelism models

30



Distinction

ESE5390:
Hardware/Software Co-Design for Machine Learning

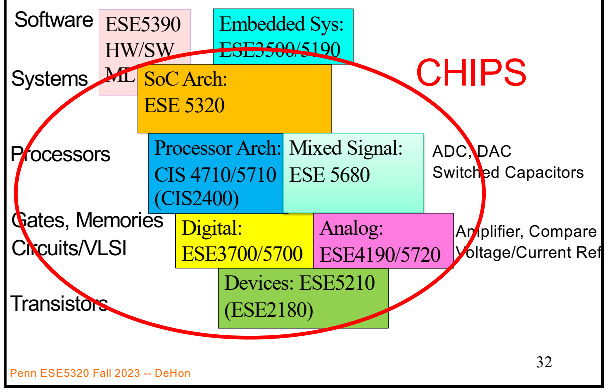
- Deep on Application (ML)
- More accessible to CS
  - Less previous experience with circuits and architecture
- Won't be as deep on understanding HW and optimization
- Program in Pytorch, OpenCL

ESE5320:

- Deep computer engineering
- Broad application
- Program in C
- Suitable followup if want to dig deeper

31

Abstraction Stack



Software ESE5390 HW/SW ML Embedded Sys: ESE3500/5190

Systems SoC Arch: ESE 5320

CHIPS

Processors Processor Arch: CIS 4710/5710 (CIS2400) Mixed Signal: ESE 5680 ADC, DAC Switched Capacitors

Gates, Memories Circuits/VLSI Digital: ESE3700/5700 Analog: ESE4190/5720 Amplifier, Compare Voltage/Current Ref

Transistors Devices: ESE5210 (ESE2180)

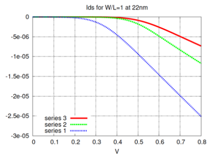
Part 3:
Approach -- Example

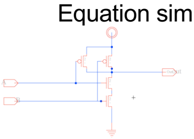
Abstract Approach

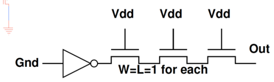
- Identify requirements, bottlenecks
- Decompose Parallel Opportunities
  - At extreme, how parallel could make it?
  - What forms of parallelism exist?
    - Thread-level, data parallel, instruction-level
- Design space of mapping
  - Choices of where to map, area-time tradeoffs
- Map, analyze, refine
  - Write equations to understand, predict

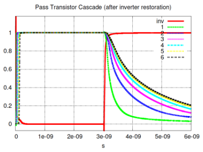
Example SPICE Circuit Simulator

- SPICE – Simulate and validate chip performance at transistor level (ESE3700,5700)
  - Specialized Differential Equation simulator for circuits (chips)

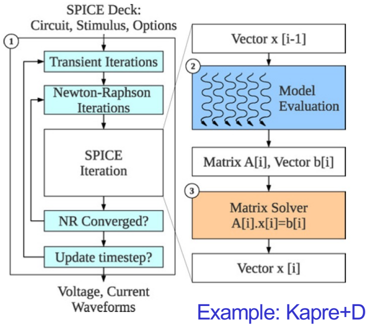








Example: SPICE Circuit Simulator



Matrix Solve
Ax=B
A matrix
B vector
x unknown vector
Solve for x
= KCL, KVL *

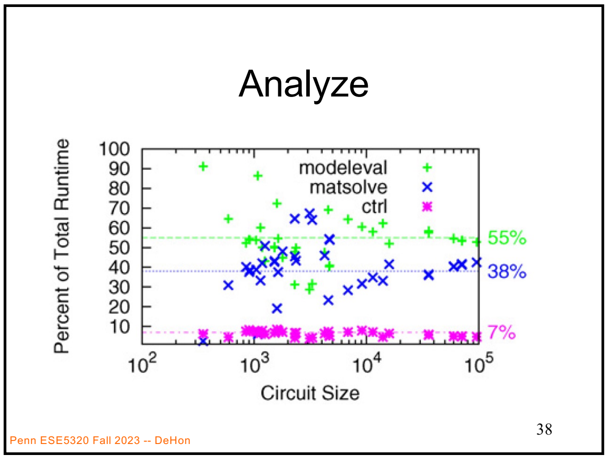
Linear Algebra
solving n eqns
in n unknowns.

Example: Kapre+DeHon, TRCAD 2012

* Kirchhoff {Current,Voltage} Laws

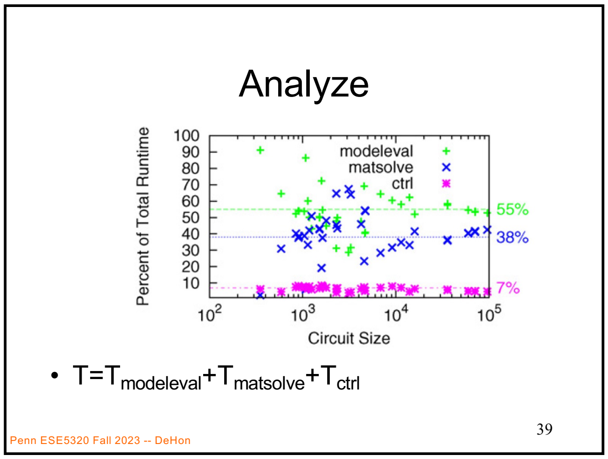
Abstract Approach

- Identify requirements, bottlenecks
- Decompose Parallel Opportunities
  - At extreme, how parallel could make it?
  - What forms of parallelism exist?
    - Thread-level, data parallel, instruction-level
- Design space of mapping
  - Choices of where to map, area-time tradeoffs
- Map, analyze, refine
  - Write equations to understand, predict



Analyze

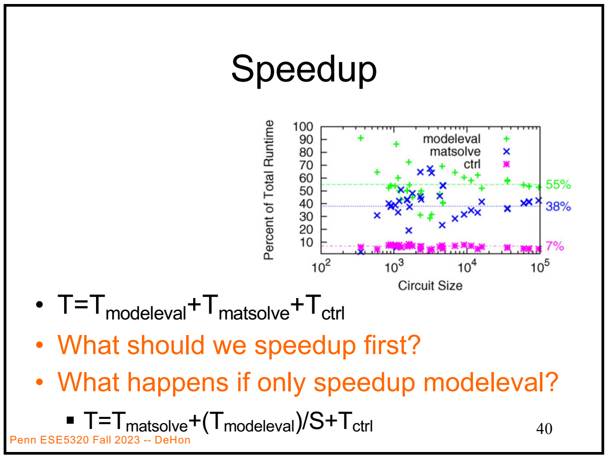
38



Analyze

- $T=T_{modeleval}+T_{matsolve}+T_{ctrl}$

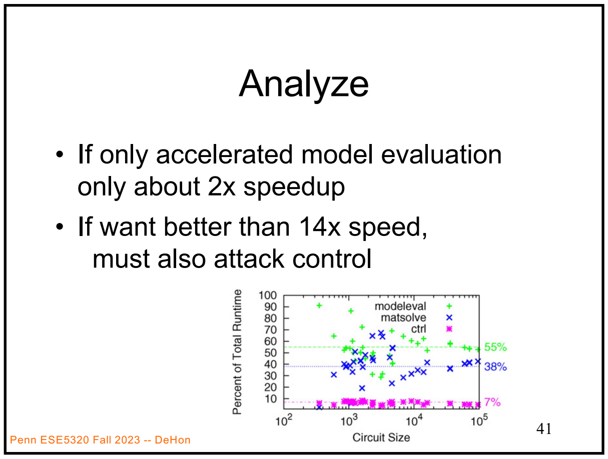
39



Speedup

- $T=T_{modeleval}+T_{matsolve}+T_{ctrl}$
- What should we speedup first?
- What happens if only speedup modeleval?
  - $T=T_{matsolve}+(T_{modeleval})/S+T_{ctrl}$

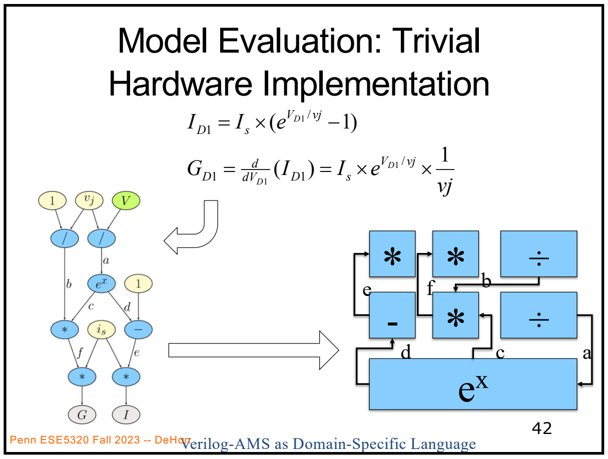
40



Analyze

- If only accelerated model evaluation only about 2x speedup
- If want better than 14x speed, must also attack control

41



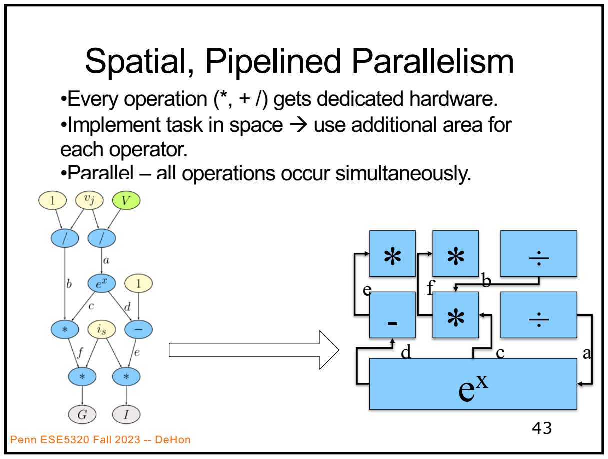
Model Evaluation: Trivial Hardware Implementation

$$I_{D1} = I_s \times (e^{V_{D1}/vj} - 1)$$

$$G_{D1} = \tfrac{d}{dV_{D1}}(I_{D1}) = I_s \times e^{V_{D1}/vj} \times \frac{1}{vj}$$

Verilog-AMS as Domain-Specific Language

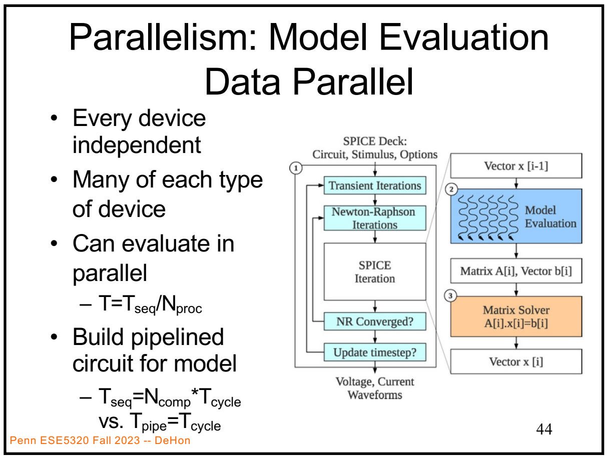
42



Spatial, Pipelined Parallelism

•Every operation (*, + /) gets dedicated hardware.
•Implement task in space → use additional area for each operator.
•Parallel – all operations occur simultaneously.

43



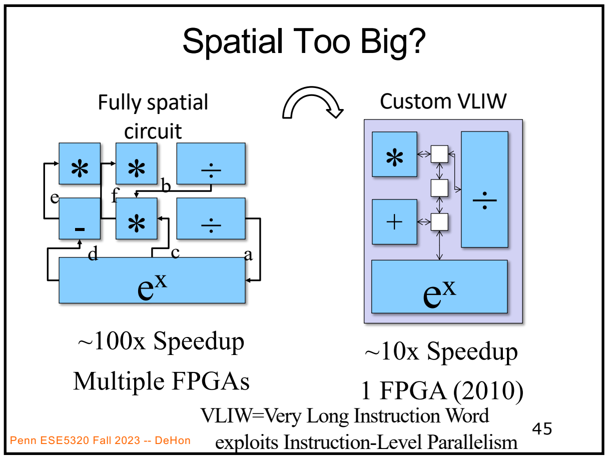
Parallelism: Model Evaluation Data Parallel

- Every device independent
- Many of each type of device
- Can evaluate in parallel
  - $T = T_{seq} / N_{proc}$
- Build pipelined circuit for model
  - $T_{seq} = N_{comp} * T_{cycle}$ vs. $T_{pipe} = T_{cycle}$

Penn ESE5320 Fall 2023 -- DeHon

44

44



Spatial Too Big?

~100x Speedup

Multiple FPGAs

~10x Speedup

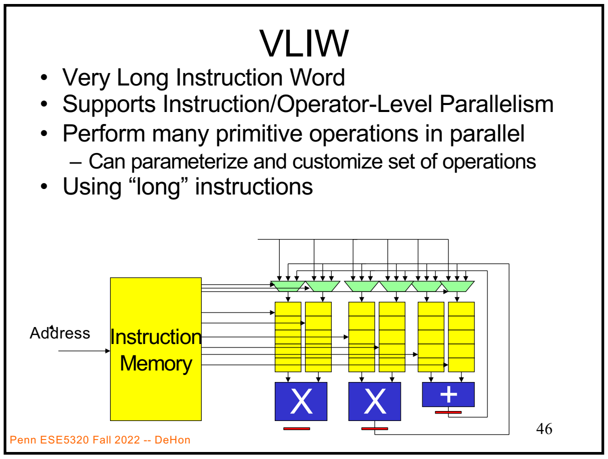
1 FPGA (2010)

VLIW=Very Long Instruction Word exploits Instruction-Level Parallelism

Penn ESE5320 Fall 2023 -- DeHon

45

45



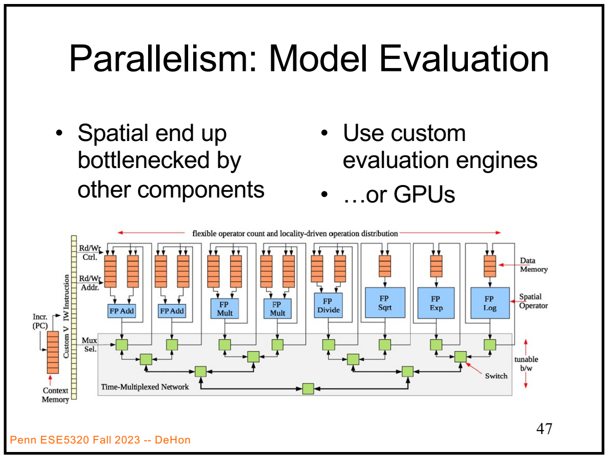
VLIW

- Very Long Instruction Word
- Supports Instruction/Operator-Level Parallelism
- Perform many primitive operations in parallel
  - Can parameterize and customize set of operations
- Using "long" instructions

Penn ESE5320 Fall 2022 -- DeHon

46

46



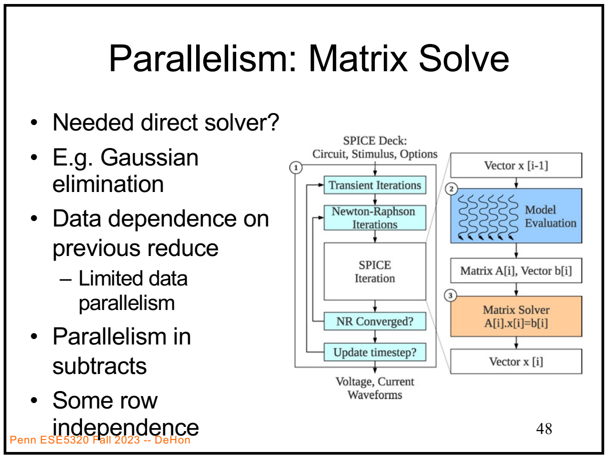
Parallelism: Model Evaluation

- Spatial end up bottlenecked by other components
- Use custom evaluation engines
- …or GPUs

Penn ESE5320 Fall 2023 -- DeHon

47

47



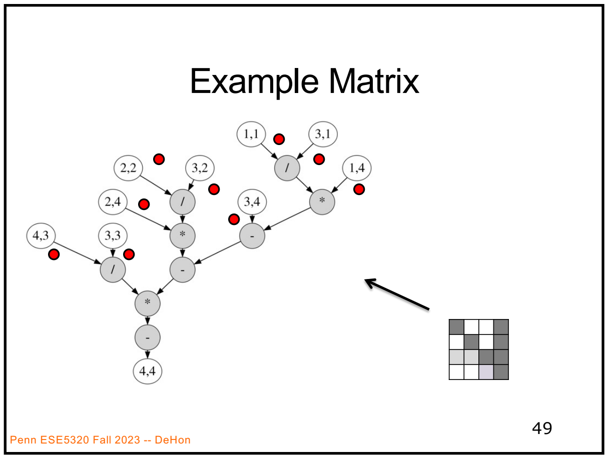
Parallelism: Matrix Solve

- Needed direct solver?
- E.g. Gaussian elimination
- Data dependence on previous reduce
  - Limited data parallelism
- Parallelism in subtracts
- Some row independence

Penn ESE5320 Fall 2023 -- DeHon

48

48

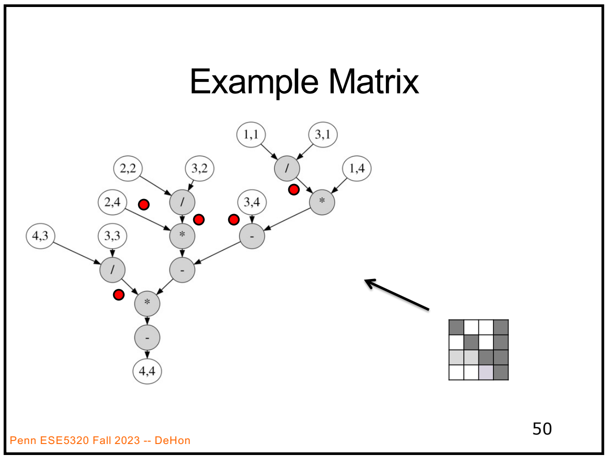


Example Matrix

Penn ESE5320 Fall 2023 -- DeHon

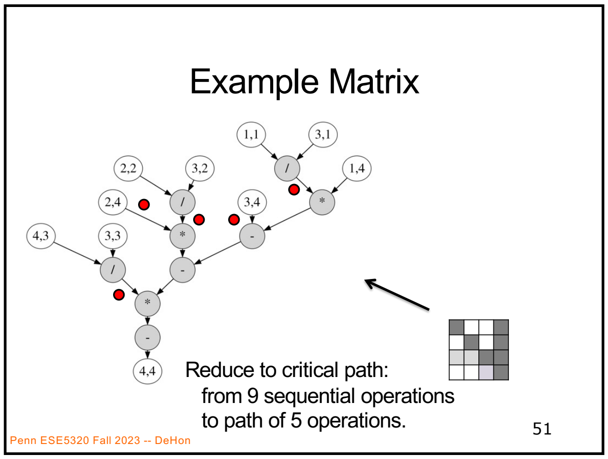
49

49



Example Matrix

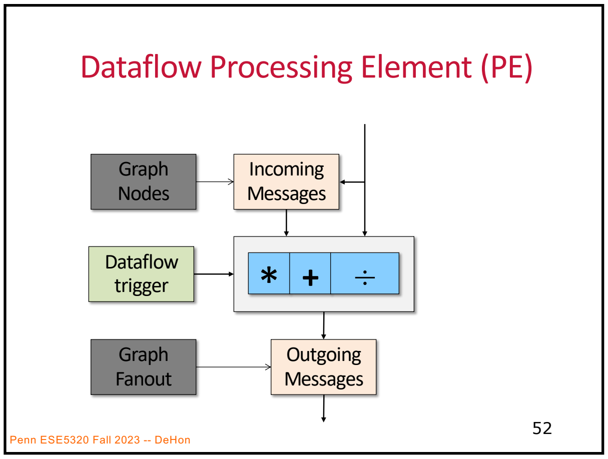
50



Example Matrix

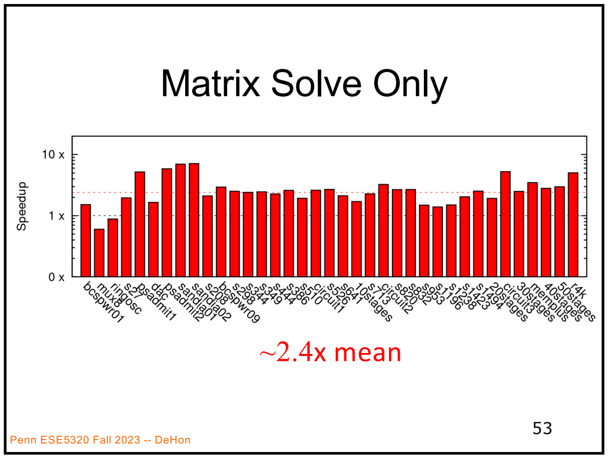
Reduce to critical path:
from 9 sequential operations
to path of 5 operations.

51



Dataflow Processing Element (PE)

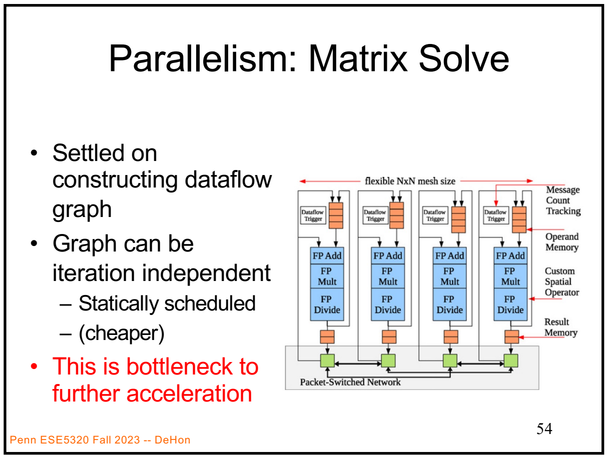
52



Matrix Solve Only

~2.4x mean

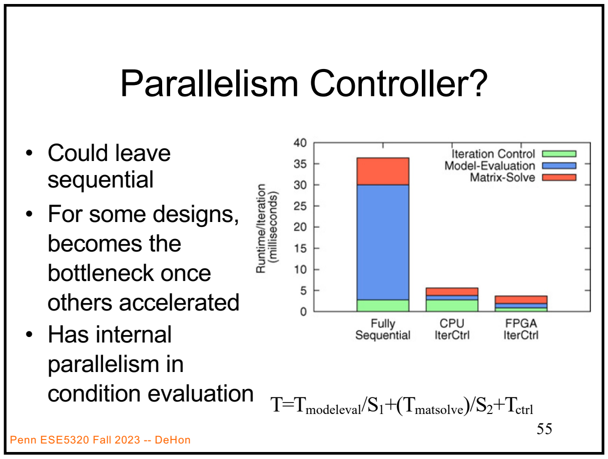
53



Parallelism: Matrix Solve

- Settled on constructing dataflow graph
- Graph can be iteration independent
  - Statically scheduled
  - (cheaper)
- This is bottleneck to further acceleration

54



Parallelism Controller?

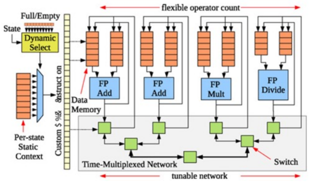
- Could leave sequential
- For some designs, becomes the bottleneck once others accelerated
- Has internal parallelism in condition evaluation

$T=T_{modeleval}/S_1+(T_{matsolve})/S_2+T_{ctrl}$

55

Parallelism Controller

- Customized datapath controller

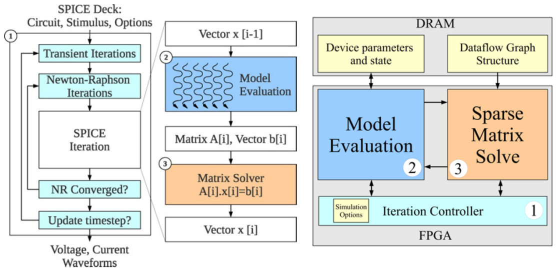


$T_{seqctrl}=N_{add}+N_{mul}+10*N_{divide}$

$T_{vliwctrl}=Max(N_{add}/2,N_{mul},10*N_{divide})$

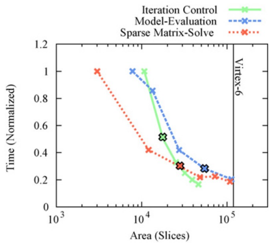
56

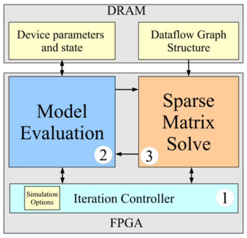
Single-Chip Solution



57

Area-Time for Each

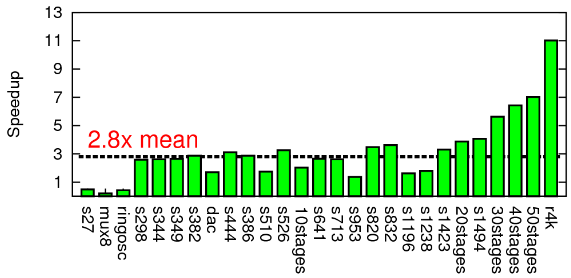




[1 Slice = 4 LUTs]

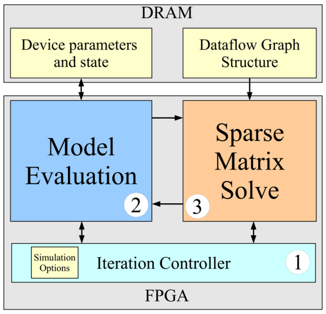
58

Composite Speedup



59

Modern SoC



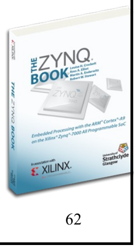
60

Part 4:
Class Components

61

Class Components

- Lecture (incl. preclass exercise)
  - In-person (not hybrid, don't expect recordings)
  - Slides on web before class (print if you want)
  - N.B. I encourage class participation
    - In class; Questions ("warm" calls)
  - Daily Quiz
- Reading [~1 required paper/lecture]
  - online: Canvas, IEEE, ACM, also ZynqBook, Parallel Programming for FPGAs
- Homework
  - (1 per week due F5pm Eastern)
- Project – open-ended (~6 weeks)
- Note syllabus, course admin online



First Half

Quickly cover breadth

- Metrics, bottlenecks
- Memory
- Parallel models
- SIMD/Data Parallel
- Thread-level parallelism
- Spatial, C-to-gates

Line up with homeworks

Second Half

- Use everything on project
- Going deeper

- Memory
- Verification
- VLIW
- Reduce
- Energy
- Chip Cost
- Real-time
- Reactive

Teaming

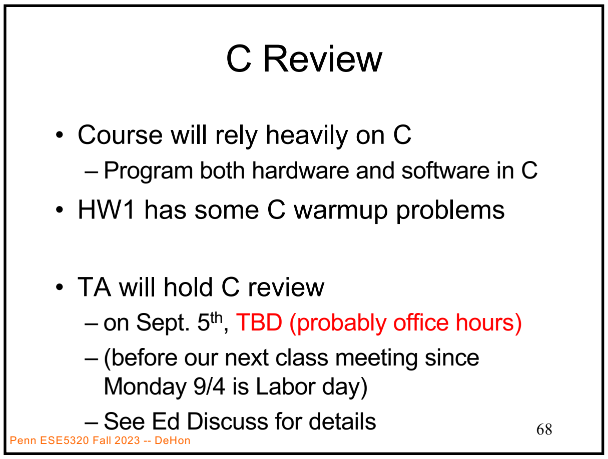
- HomeWorks (HW) in Groups of 2 (after 0, 1)
- HW: we assign
- Individual assignment writeup
- Project in Groups of 3
- Project: you propose team of 3, we review
  - Most portions group writeup
  - Maybe few components individual writeup

Office & Lab Hours

- Andre: M 4:00pm—5:00pm
  - Levine 270, Zoom
  - See canvas
- TAs – Ketterer (starting next week)
  - Tuesday 6 pm
  - Wednesday 7 pm (not today)
  - Thursday 8:15pm-9:15pm (not tomorrow)

Diagnostic Assessment

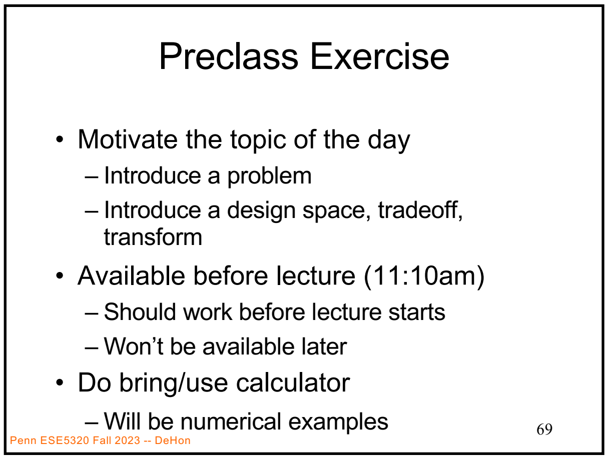
- Course will rely heavily on C
  - Program both hardware and software in C
- If you cannot read/write code in C, this course will be a challenge
- Diagnostic Assessment intended as a quick indication if you aren't ready
  - Should be able to complete quickly
  - Better to find out now than after you're stuck in the course
  - Due next Wednesday (9/6)



C Review

- Course will rely heavily on C
  - Program both hardware and software in C
- HW1 has some C warmup problems
- TA will hold C review
  - on Sept. 5th, TBD (probably office hours)
  - (before our next class meeting since Monday 9/4 is Labor day)
  - See Ed Discuss for details

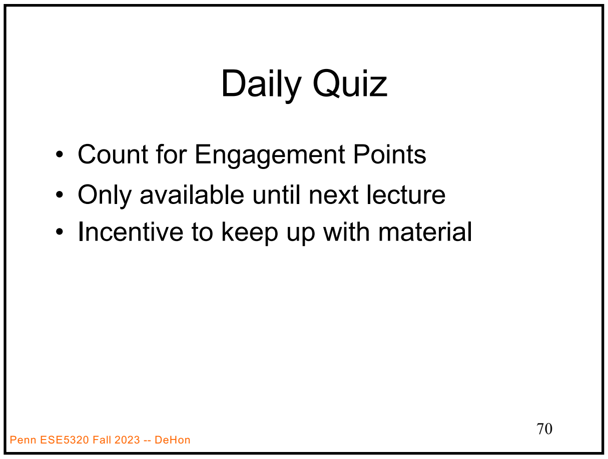
68



Preclass Exercise

- Motivate the topic of the day
  - Introduce a problem
  - Introduce a design space, tradeoff, transform
- Available before lecture (11:10am)
  - Should work before lecture starts
  - Won't be available later
- Do bring/use calculator
  - Will be numerical examples

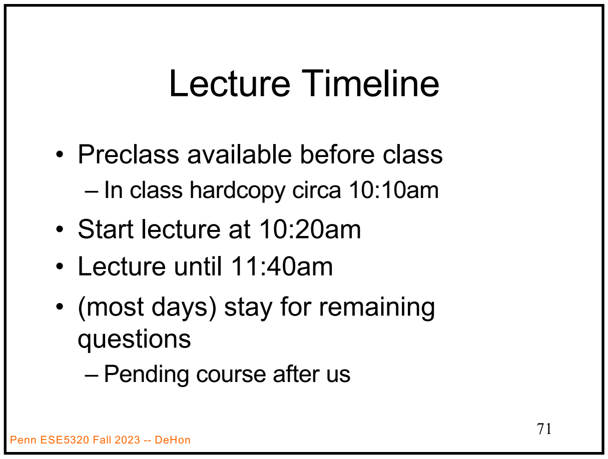
69



Daily Quiz

- Count for Engagement Points
- Only available until next lecture
- Incentive to keep up with material

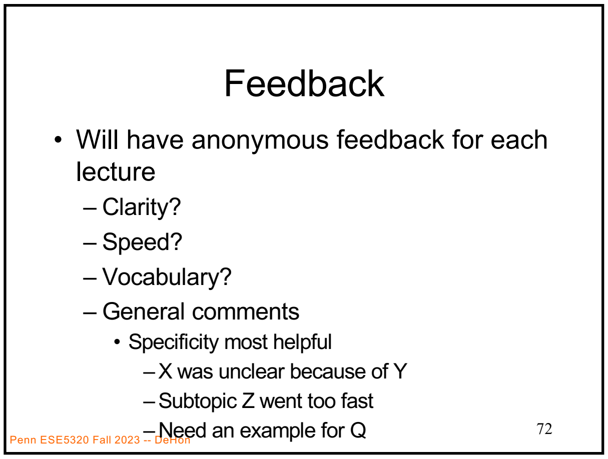
70



Lecture Timeline

- Preclass available before class
  - In class hardcopy circa 10:10am
- Start lecture at 10:20am
- Lecture until 11:40am
- (most days) stay for remaining questions
  - Pending course after us

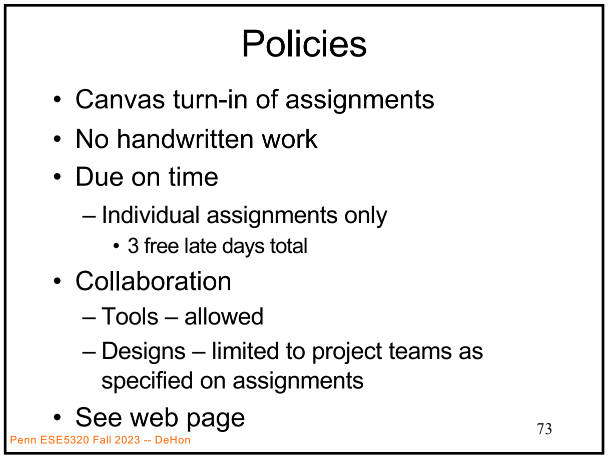
71



Feedback

- Will have anonymous feedback for each lecture
  - Clarity?
  - Speed?
  - Vocabulary?
  - General comments
    - Specificity most helpful
      - X was unclear because of Y
      - Subtopic Z went too fast
      - Need an example for Q

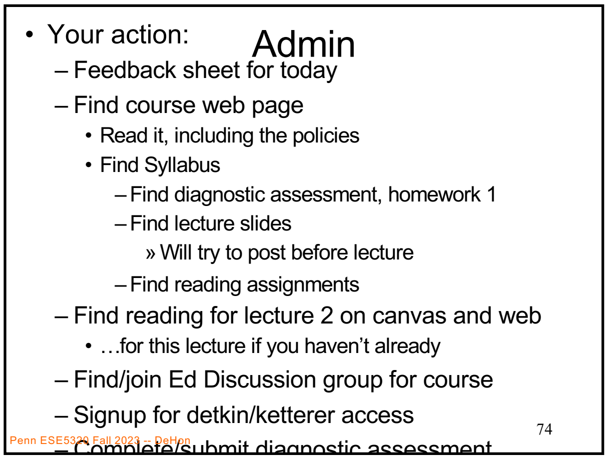
72



Policies

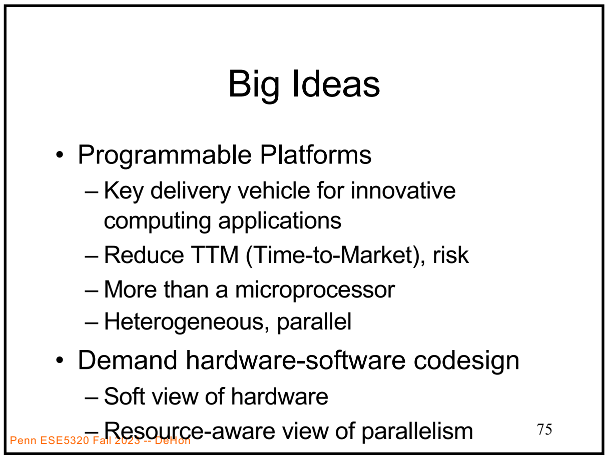
- Canvas turn-in of assignments
- No handwritten work
- Due on time
  - Individual assignments only
    - 3 free late days total
- Collaboration
  - Tools – allowed
  - Designs – limited to project teams as specified on assignments
- See web page

73



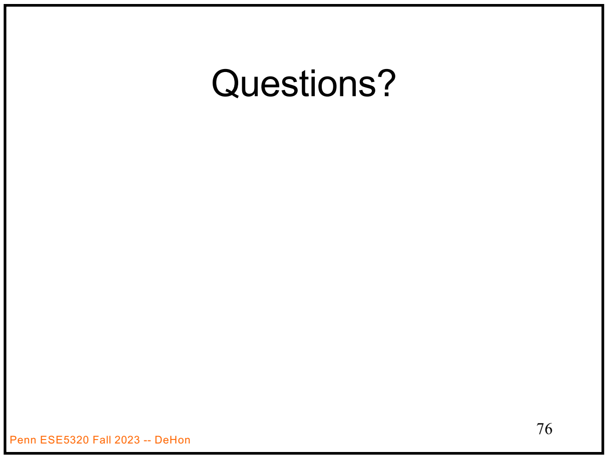
Admin

- Your action:
  - Feedback sheet for today
  - Find course web page
    - Read it, including the policies
    - Find Syllabus
      - Find diagnostic assessment, homework 1
      - Find lecture slides
        - » Will try to post before lecture
      - Find reading assignments
  - Find reading for lecture 2 on canvas and web
    - …for this lecture if you haven't already
  - Find/join Ed Discussion group for course
  - Signup for detkin/ketterer access
  - Complete/submit diagnostic assessment



Big Ideas

- Programmable Platforms
  - Key delivery vehicle for innovative computing applications
  - Reduce TTM (Time-to-Market), risk
  - More than a microprocessor
  - Heterogeneous, parallel
- Demand hardware-software codesign
  - Soft view of hardware
  - Resource-aware view of parallelism



Questions?