ESE532: System-on-a-Chip Architecture

Day 1: September 1, 2010
Introduction and Overview
(lecture start target 10:20am)

Note: both preclass and feedback linked to web (also slides)
www.seas.upenn.edu/~ese532/fall2021/fall2021.html
- Preclass
- Feedback form

Today

- Part 1: Case for Programmable SoC
- Part 2
  - Course Goals
  - Outcomes
  - Evovling Course, Risks, Tools
- Part 3: Sample Optimization
- Part 4: This course
  - (incl. policies, logistics)

Apple A14 Bionic

- 88mm^2, 5nm
- 11.8 Billion Tr.
- iPhone 12
- 6 ARM cores
  - 2 fast (2.9--3GHz)
  - 4 low energy
- 4 custom GPUs
- 16 Neural Engines
  - 11 Trillion ops/s?

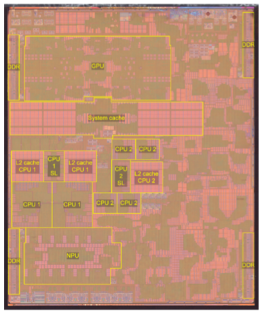
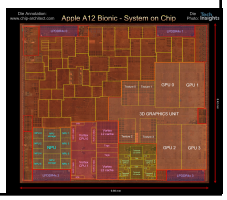


Image from https://www.extremetech.com/computing/318715-comparison-of-apple-m1-a14-shows-differences-in-soc-design
details: https://www.tomshardware.com/news/apple-a14-bionic-revealed
https://www.anandtech.com/show/16226/apple-silicon-m1-a14-deep-dive/2

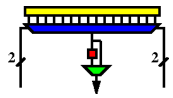
Questions

- 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) …. ?

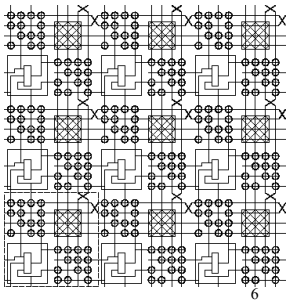


FPGA Field-Programmable Gate Array

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



ESE150, CIS571

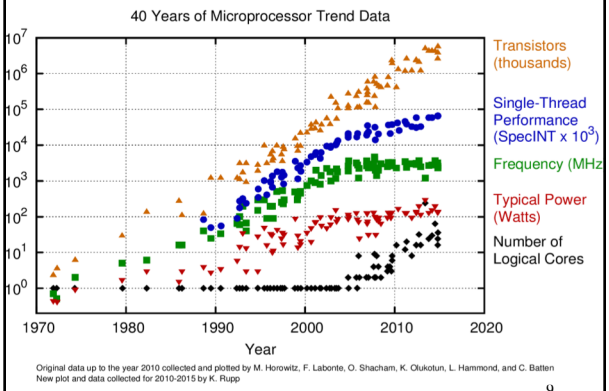


Case for Programmable SoC

End of uProcessor 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
- uP were burning more power
- Lost ability to scale down voltage
- Processor performance stalled



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 IC
- Wanted product differentiation
  - Got it at the board level
  - Select which ICs and how wired
- Build a custom IC
  - 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 Billions of Transistors

Non-Recurring Engineering (NRE) Costs

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

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

NRE Costs

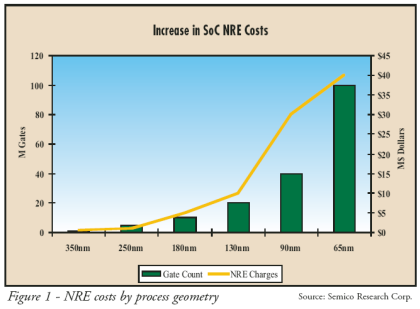
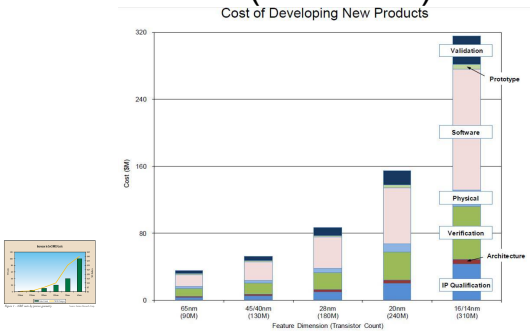


Figure 1 - NRE costs by process geometry Source: Semico Research Corp.

NRE Cost (continued)



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

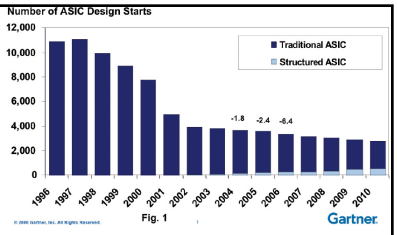
Amortize NRE with Volume

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

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

Economics

Forcing fewer, more customizable chips



- Economics force fewer, more customizable chips
  - Mask costs in the millions of dollars
  - Custom IC design NRE 10s—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

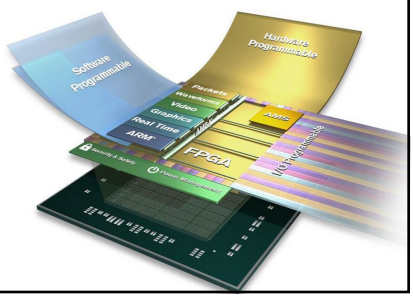
- Now contain significant software
  - Almost all have embedded processors
- Must co-design SW and HW
- Must solve complete computing task
  - Tasks has components with variety of needs
  - Some don't need custom circuit
  - 90/10 Rule

Given Demand for Programmable

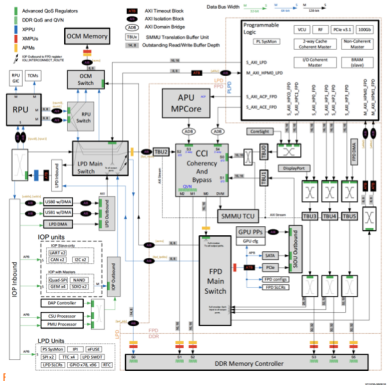
- How do we get higher performance than a processor, while retaining programmability?

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
  - 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
- Wanted product differentiation
  - Program FPGAs, PSoC
- Build a custom IC
  - 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
- 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

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.

Evolving Course

- Spring 2017 – first offering
  - Raw, all assignments new, buggy, too tedious, long
- Fall 2017 – second offering
  - Refine assignments, project; increased explicit modeling emphasis
  - Hard, not insane
- Fall 2018 – third offering (similar 2017)
  - Added real-time ethernet data handling; project groups of 3
  - Many students challenged with C and software engineering
  - Stream debug and performance challenging
- Fall 2019 – fourth (structure same)
  - Try front-load more C, better introduce Stream optimization and debug
  - Group writeup on projects

Evolving Course

- Fall 2018 – third offering (similar 2017)
  - Added real-time ethernet data handling; project groups of 3
  - Many students challenged with C and software engineering
  - Stream debug and performance challenging
- Fall 2019 – fourth (structure same)
  - Try front-load more C, better introduce Stream optimization and debug
  - Group writeup on projects
- Fall 2020 – fifth (structure same)
  - Move to Vitis (from SDSoC)
  - Use Amazon cloud for first half; F1 instance for FPGA access HW
  - Then transition to Ultra96 (SoC FPGA) for projects
- Fall 2021 – sixth
  - Stay with Vitis; use DFX (see next slide)
  - Ultra96; no Amazon cloud
  - Introduce project components earlier

Tools

- Are complex
- Will be challenging, but good for you to build confidence can understand and master
- Tool runtimes can be long
  - Maybe DFX will help
  - DFX – Dynamic Function Exchange
    - Partial reconfiguration
- Learning and sharing experience will be part of assignments

Distinction

CIS240, 371, 471, 571

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

ESE532

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

Distinction

ESE539:

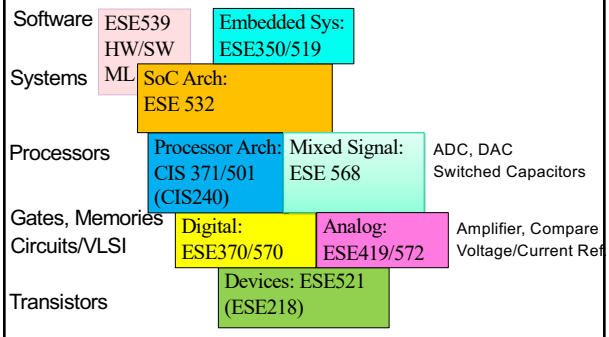
Hardware/Software Co-Design for Machine Learning

- 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

ESE532:

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

Abstraction Stack

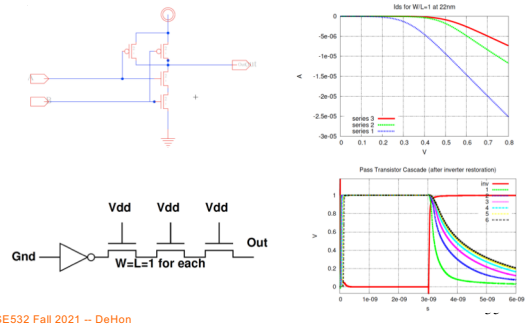


Part 3: Approach -- Example

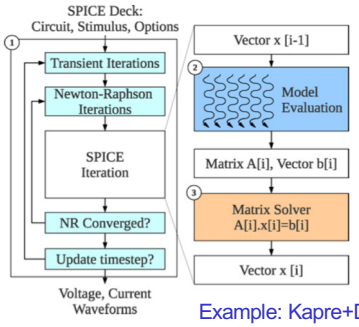
Abstract Approach

- 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



Example: SPICE Circuit Simulator



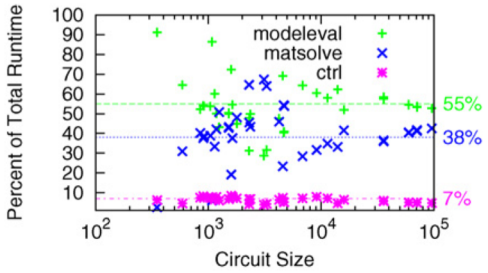
Matrix Solve
Ax=B
A matrix
B vector
x unknown vector
Solve for x
= KCL, KVL *

Linear Algebra
solving n eqns
in n unknowns.

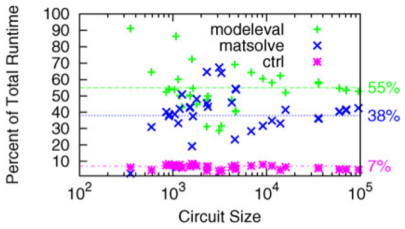
Example: Kapre+DeHon, TRCAD 2012

* Kirchhoff {Current,Voltage} Laws

Analyze

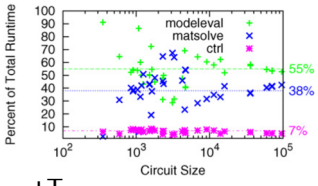


Analyze



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

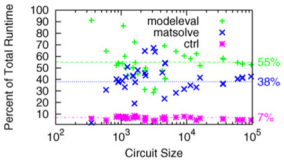
Speedup



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

Analyze

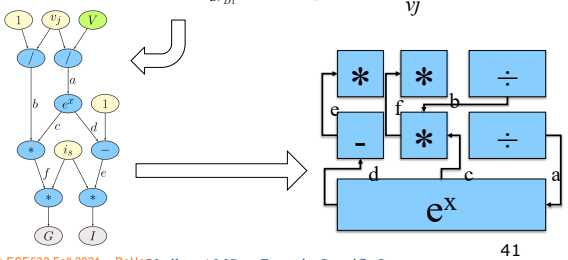
- If only accelerated model evaluation only about 2x speedup
- If want better than 14x speed, must also attack control



Model Evaluation: Trivial Hardware Implementation

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

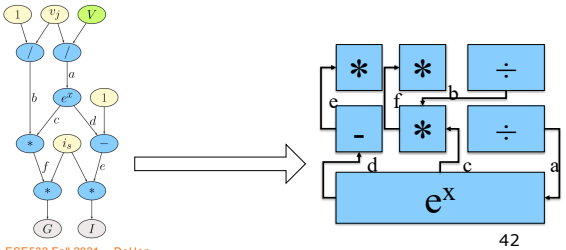
$$G_{D1} = \frac{d}{dV_{D1}}(I_{D1}) = I_s \times e^{V_{D1}/vj} \times \frac{1}{vj}$$



Verilog-AMS as Domain-Specific Language

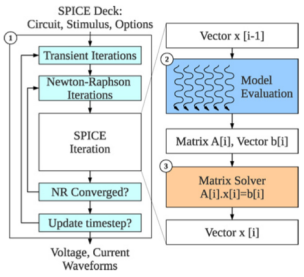
Spatial, Pipelined Parallelism

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

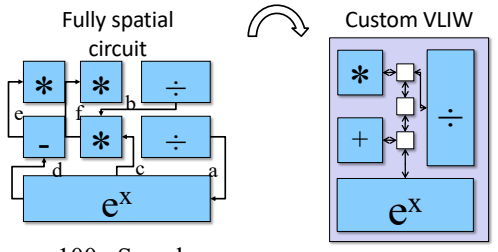


Parallelism: Model Evaluation Data Parallel

- 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. $T_{pipe}=T_{cycle}$



Spatial Too Big?



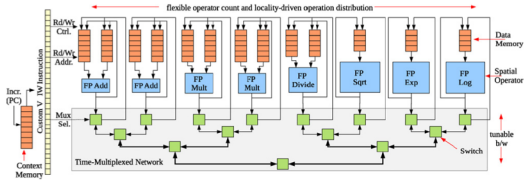
~100x Speedup
Multiple FPGAs

~10x Speedup
1 FPGA

VLIW=Very Long Instruction Word
exploits Instruction-Level Parallelism

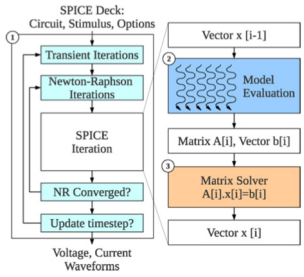
Parallelism: Model Evaluation

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

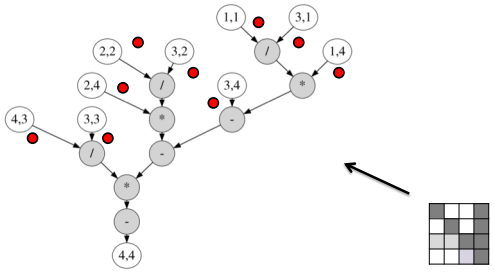


Parallelism: Matrix Solve

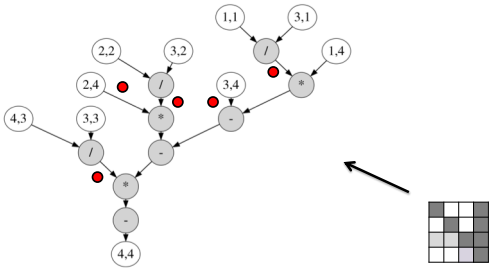
- Needed direct solver?
- E.g. Gaussian elimination
- Data dependence on previous reduce
  - Limited data parallelism
- Parallelism in subtracts
- Some row independence



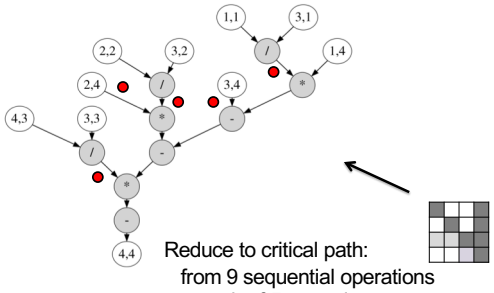
Example Matrix



Example Matrix

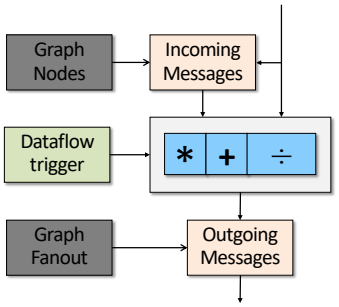


Example Matrix

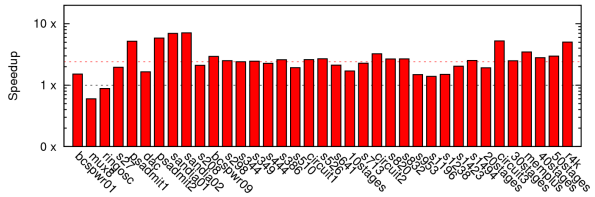


Reduce to critical path:
from 9 sequential operations
to path of 5 operations.

Dataflow Processing Element (PE)



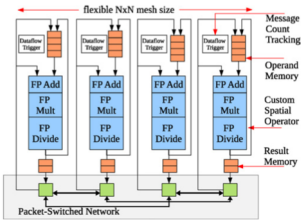
Matrix Solve Only



~2.4x mean

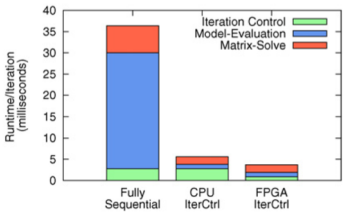
Parallelism: Matrix Solve

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



Parallelism Controller?

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



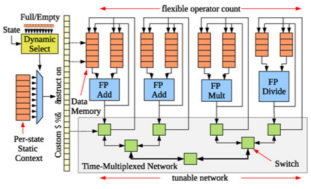
$T=T_{modeleval}/S_1+(T_{matsolve})/S_2+T_{ctrl}$

Parallelism Controller

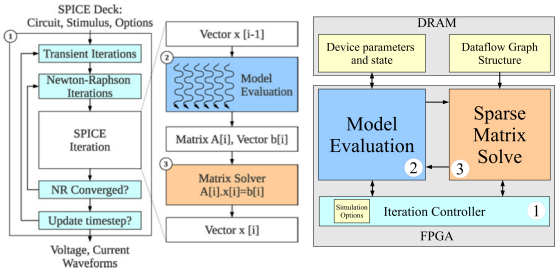
- Customized datapath controller

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

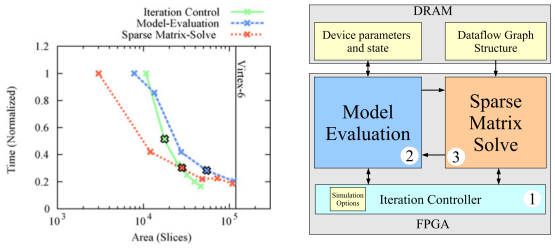
$T_{vliwctrl}=Max(N_{add}/2,N_{mul},10*N_{divide})$



Single-Chip Solution

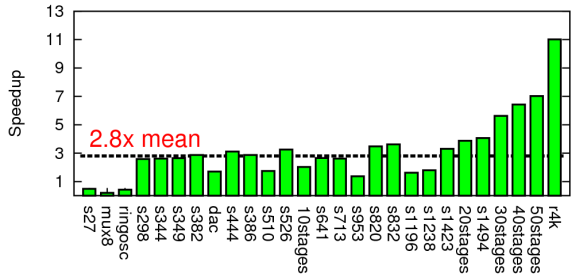


Area-Time for Each

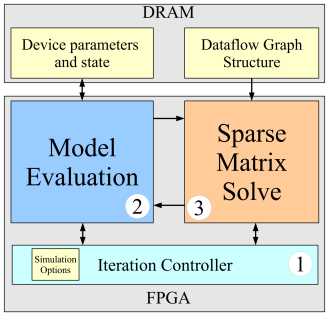


[1 Slice = 4 LUTs]

Composite Speedup



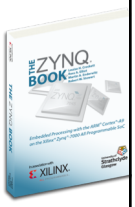
Modern SoC



Part 4: Class Components

Class Components

- Lecture (incl. preclass exercise)
  - Slides on web before class
    - (you can print if want a follow-along copy)
  - N.B. I encourage class participation
    - In class/synch recording; 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
- Schedule more tentative
  - Adjust as experience and project demands
- Going deeper
- Real-time
- Reactive
- Memory
- Networking
- Energy
- Scaling
- Chip Cost
- Verification

Teaming

- HW in Groups of 2
- HW: we assign
- Individual assignment writeup
- Project in Groups of 3
- Project: you propose, we review
  - Most portions group writeup
  - Few components individual writeup

Office & Lab Hours

- Andre: T 4:15pm—5:30pm
  - Levine 270, Zoom
  - See canvas
- TAs -- Detkin
  - Tuesday 5 pm
  - Wednesday 5 pm (not today)
  - Thursday 5—7pm (first office hours tomorrow)

C Review

- Course will rely heavily on C
  - Program both hardware and software in C
- HW1 has some C warmup problems
- TAs will hold C review
  - on Sept. 7th, 5:00pm
  - (before our next class meeting since Monday 9/6 is Labor day)

Preclass Exercise

- Motivate the topic of the day
  - Introduce a problem
  - Introduce a design space, tradeoff, transform
- Available before lecture (and in lecture)
  - (only available for 24-48 hours; download)
  - Should work before lecture starts
- Do bring/use calculator
  - Will be numerical examples

Diagnostic Quiz

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

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
- Post video to canvas later in day

Feedback

- Will have anonymous feedback {paper, google forms} for each lecture
  - Clarity?
  - Speed?
  - Vocabulary?
  - General comments
- Paper hardcopy for in-person
- Linked on syllabus for not in-person:
  - https://forms.gle/FWdNWjCsnv6F4pGf8

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

Hybrid

- Uncertain how hybrid zoom/in-person will work
  - Setup a bit of a challenge
  - Interactive work?

Admin

- Your action:
  - Find course web page
    - Read it, including the policies
    - Find Syllabus
      - Find 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 piazza group for course
  - Signup for detkin/ketterer access

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?