ESE532: System-on-a-Chip Architecture

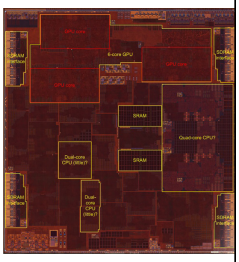
Day 1: January 11, 2017
Introduction and Overview

Today

- Case for Programmable SoC
- Goals
- Outcomes
- New Course, Risks, Tools
- Sample Optimization
- This course (incl. policies, logistics)
- Zed Boards

Today SoC: Apple A10

- Quad=Dual Dual Core
  - Dual 64-bit ARM 2.3GHz
  - Dual 64-bit ARM low energy
- 3MB L2 cache
- 6 GPU cores
- Custom accelerators
  - Image Processor?
- 125mm^2 16nm FinFET
- 3.3B transistors



Chipworks Die Photo

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) …. ?

Case for Programmable SoC

The Way things Were

20 years ago

- Wanted programmability
  - used a processor
- 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
- 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

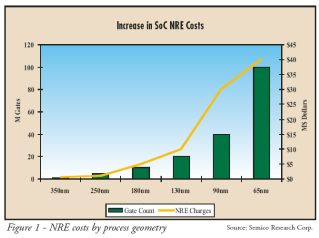


Figure 1 - NRE costs by process geometry Source: Semico Research Corp.

- **28-nm SoC development costs doubled over previous node – EE Times 2013**

28nm+78%, 20nm+48%, 14nm+31%, 10nm+35%

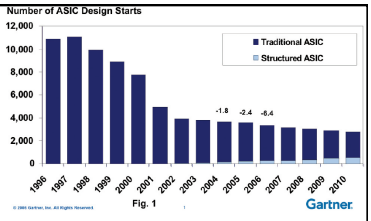
Amortize NRE with Volume

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

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

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

Large ICs

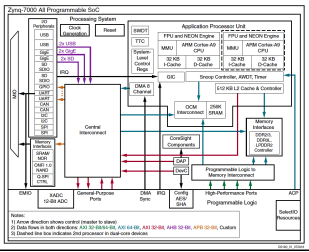
- Now contain significant software
  - Almost all have embedded processors
- Must co-design SW and HW
- Must solve complete computing task
  - Tasks has components with variety of needs
  - Some don't need custom circuit
  - 90/10 Rule

Given Demand for Programmable

- How get beyond processors?
- Multiple processors?
- Integrate common board functionality?
  - Memory, communications, I/O, peripherals
- Accelerators?
  - Generalized
  - Specialized (for what?)

Programmable SoC

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



Goals, Outcomes

Goals

- Create Computer Engineers
  - SW/HW divide is wrong, outdated
  - 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
- 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, RTL 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.

New Course

- We'll be making it up as we go along…
- You'll be first to perform assignments
- We may estimate difficulty of assignments incorrectly
  - Too easy, too hard
- Lectures will be less polished
- Intellectual excitement of trying to figure it out…and currency

Risks

- Tools tricky to get working
  - New SDSoC
  - New try use HLS (C-to-gates) throughout
- Tool runtimes can be long
- Choosing problems with
  - Reasonable scope
  - Interesting/rich opportunities for optimization

Tools

- Are complex
- We'll be figuring them out together
- View as collaboratively figuring out how to use them this term
- Will be challenging, but good for you to build confidence can understand and master
- Learning and sharing experience will be part of assignments
  - Bonus points for tutorials

Relation to ESE532

534 – (last term)

- Deep into design space and continuum
- How to build compute, interconnect, memory
- Analysis
- Fundamentals
  - Theory
  - Why X better than Y
- More relevant substrate designers
- Both Real-Time and Best-effort

532 – System-on-a-Chip Architecture

- New course
- Probably 33% overlap
- Broader (all CMPE)
  - HW/SW codesign
- More Hands-on
  - Code in C
  - Map to Zynq
  - Accelerate an application
- More relevant to (P)SoC user
- Real-time focus

Distinction

CIS240, 371 ,501

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

ESE532

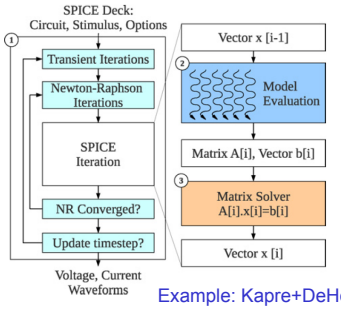
- Hardware-Software codesign
  - Willing to recompile, maybe rewrite code
  - Define/refine hardware
- Real-Time
  - Guarantee meet deadline
- Non shared-memory models

Approach -- Example

Abstract Approach

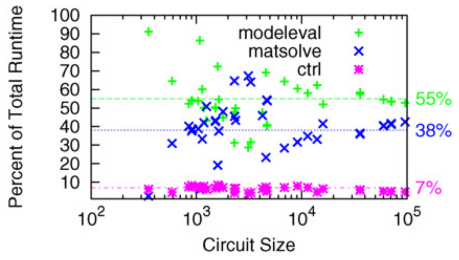
- Identify requirements, bottlenecks
- Decompose Parallel Opportunities
  - At extreme, how parallel could make it?
  - What forms of parallelism exist?
    - Thread-level, data parallel, instruction-level
- Design space of mapping
  - Choices of where to map, area-time tradeoffs
- Map, analyze, refine

SPICE Circuit Simulator



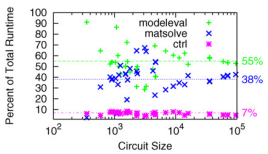
Example: Kapre+DeHon, TRCAD 2012

Analyze



Analyze

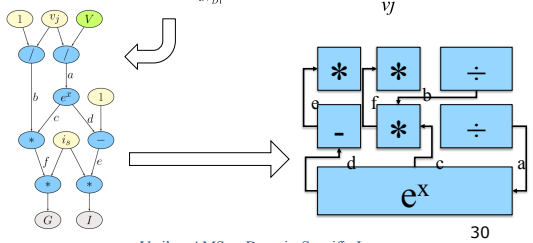
- If only accelerated model evaluation only about 2x speedup
- If want better than 14x speed, must also attack control



Model Evaluation: Trivial Hardware Implementation

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

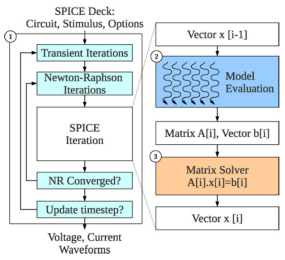
$$G_{D1} = \frac{d}{dV_{D1}}(I_{D1}) = I_s \times e^{V_{D1}/vj} \times \frac{1}{vj}$$



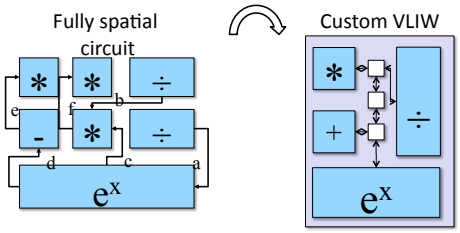
Verilog-AMS as Domain-Specific Language

Parallelism: Model Evaluation

- Every device independent
- Many of each type of device
- Can evaluate in parallel
- Could build a pipelined circuit for each model
  - Multiple if merited



Single FPGA Mapping

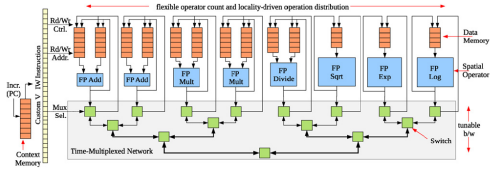


~100x Speedup
Multiple FPGAs

1-10x Speedup
1 FPGA

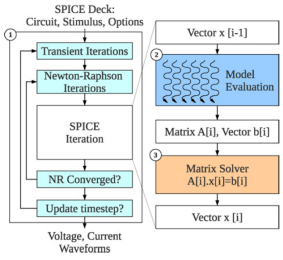
Parallelism: Model Evaluation

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

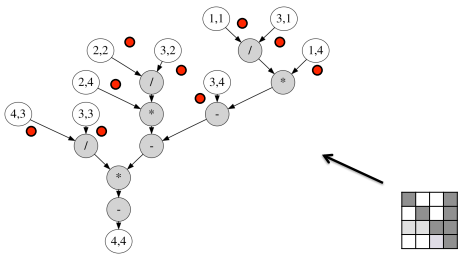


Parallelism: Matrix Solve

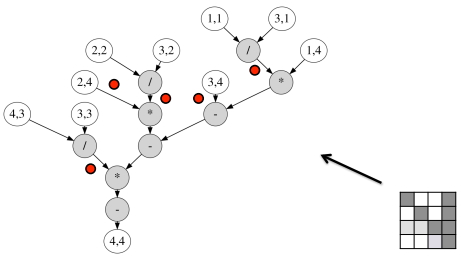
- Needed direct solver?
- E.g. Gaussian elimination
- Data dependence on previous reduce
- Parallelism in subtracts
- Some row independence



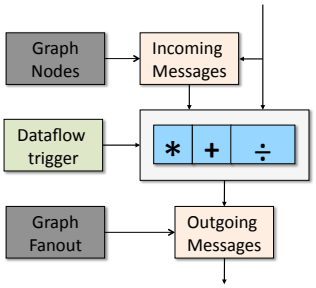
Example Matrix



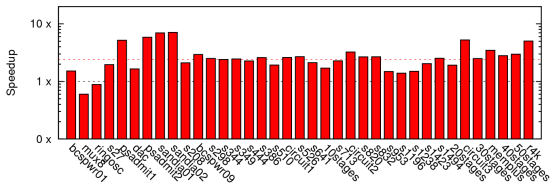
Example Matrix



Dataflow PE



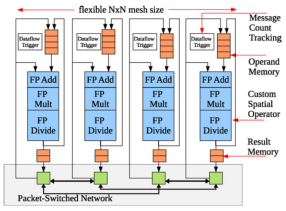
Matrix Solve Only



~2.4x mean

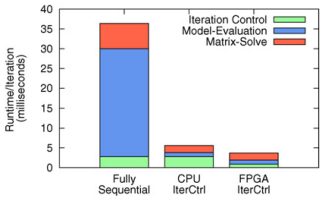
Parallelism: Matrix Solve

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



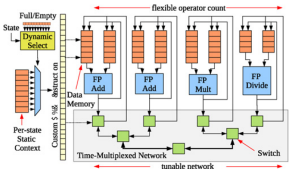
Parallelism Controller?

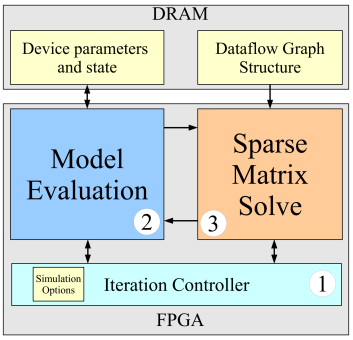
- Could leave sequential
- For some designs, becomes the bottleneck once others accelerated
- Has internal parallelism in condition evaluation



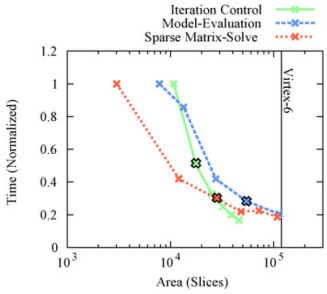
Parallelism Controller

- Customized datapath controller

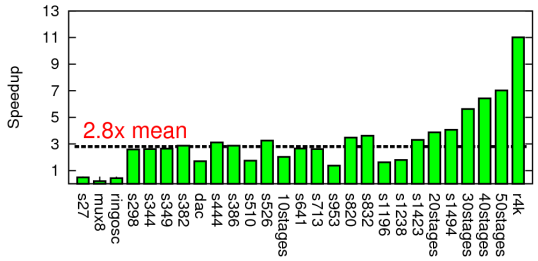




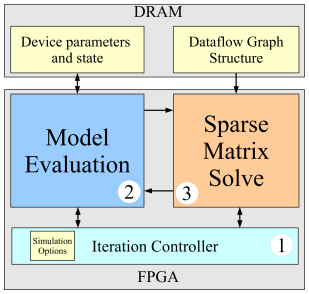
Area-Time for Each



Composite Speedup



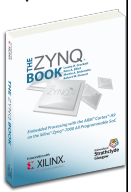
Modern SoC



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 will encourage (force) class participation
- Reading [~1 required paper/lecture]
  - online: Canvas, IEEE, ACM, also ZynqBook
- Homework
  - (1 per week up to Spring Break, due F5pm)
- Project – open-ended
  - (~6 weeks)
- Note syllabus, course admin online



First Half

Quickly cover breadth
- Metrics, bottlenecks
- Parallel models
- SIMD/Data Parallel
- Thread-level parallelism
- DMA, Memory
- Spatial, C-to-gates
- Real-time
- Reactive

Line up with homeworks

Second Half

- Use everything on project
- Schedule more tentative
  - Adjust as experience and project demands
- Going deeper
- Memory
- Networking
- Energy
- Scaling
- Chip Cost
- Defect + Fault tolerance

Teaming

- HW and Project in Groups of 2
- First assignment
  - You choose, optionally individual
- HW2-7: we assign
- Project: you propose, we review
- Individual assignment turnin
  - See assignment details if individual pieces

Office & Lab Hours

- Andre: T 4:15pm—5:30pm Levine 270
- Hans: TR 6—7pm in Ketterer
  - Start tomorrow 1/12

Preclass Exercise

- Motivate the topic of the day
  - Introduce a problem
  - Introduce a design space, tradeoff, transform
- Work for ~5 minutes before start lecturing

Feedback

- Will have anonymous feedback sheets for each lecture
  - Clarity?
  - Speed?
  - Vocabulary?
  - General comments

Policies

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

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

Big Ideas

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

Zedboard Checkout

- Ketterer Combo