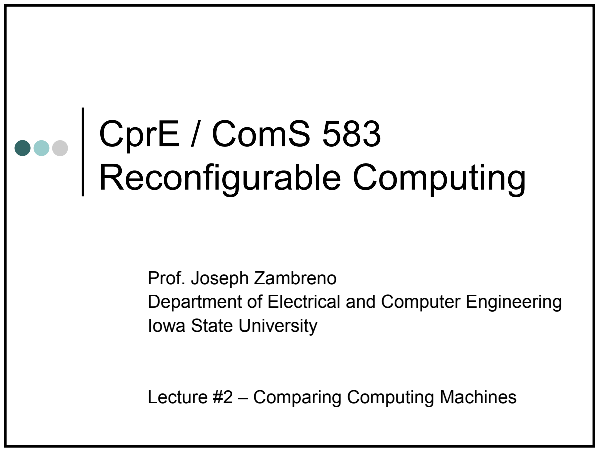
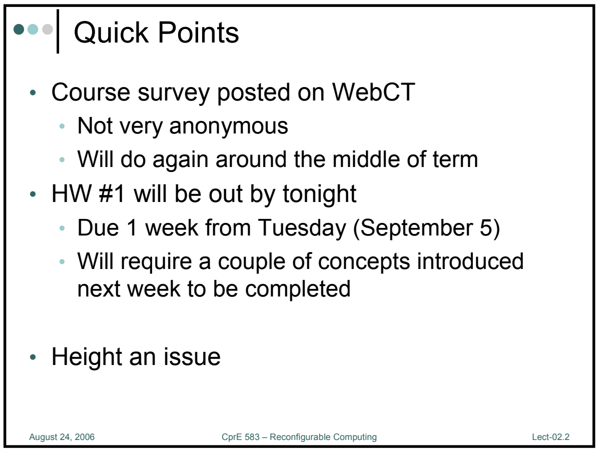


CprE / ComS 583 Reconfigurable Computing

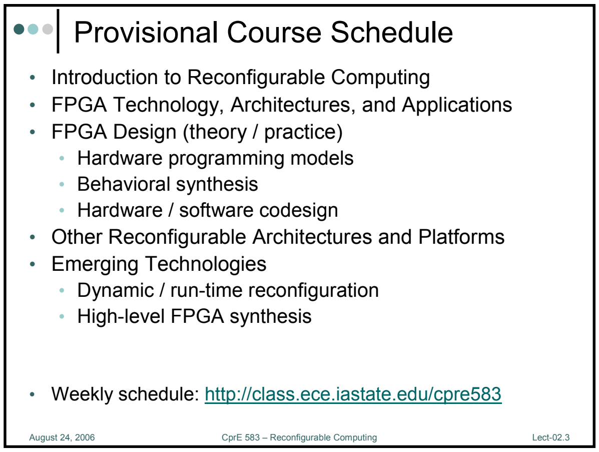
Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

Lecture #2 – Comparing Computing Machines



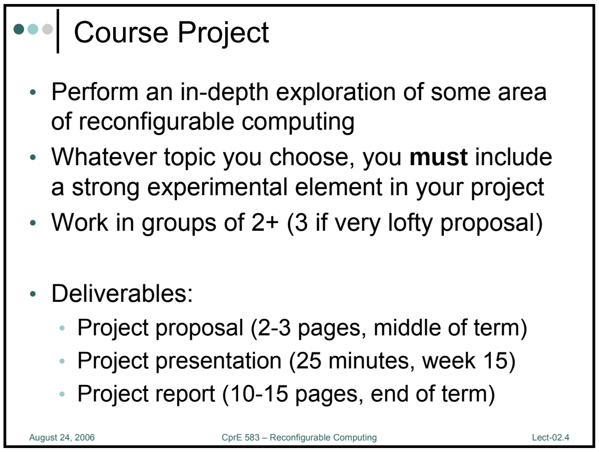
Quick Points

- Course survey posted on WebCT
  - Not very anonymous
  - Will do again around the middle of term
- HW #1 will be out by tonight
  - Due 1 week from Tuesday (September 5)
  - Will require a couple of concepts introduced next week to be completed
- Height an issue



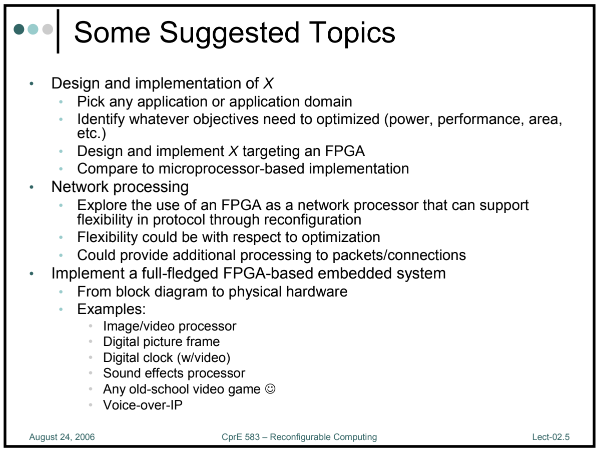
Provisional Course Schedule

- Introduction to Reconfigurable Computing
- FPGA Technology, Architectures, and Applications
- FPGA Design (theory / practice)
  - Hardware programming models
  - Behavioral synthesis
  - Hardware / software codesign
- Other Reconfigurable Architectures and Platforms
- Emerging Technologies
  - Dynamic / run-time reconfiguration
  - High-level FPGA synthesis
- Weekly schedule: http://class.ece.iastate.edu/cpre583



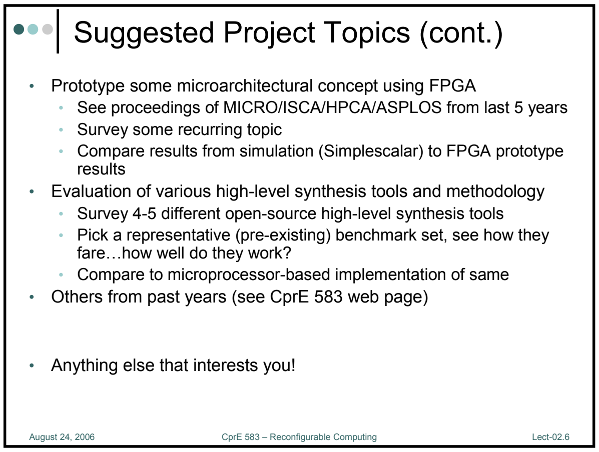
Course Project

- Perform an in-depth exploration of some area of reconfigurable computing
- Whatever topic you choose, you **must** include a strong experimental element in your project
- Work in groups of 2+ (3 if very lofty proposal)
- Deliverables:
  - Project proposal (2-3 pages, middle of term)
  - Project presentation (25 minutes, week 15)
  - Project report (10-15 pages, end of term)



Some Suggested Topics

- Design and implementation of *X*
  - Pick any application or application domain
  - Identify whatever objectives need to optimized (power, performance, area, etc.)
  - Design and implement *X* targeting an FPGA
  - Compare to microprocessor-based implementation
- Network processing
  - Explore the use of an FPGA as a network processor that can support flexibility in protocol through reconfiguration
  - Flexibility could be with respect to optimization
  - Could provide additional processing to packets/connections
- Implement a full-fledged FPGA-based embedded system
  - From block diagram to physical hardware
  - Examples:
    - Image/video processor
    - Digital picture frame
    - Digital clock (w/video)
    - Sound effects processor
    - Any old-school video game ☺
    - Voice-over-IP



Suggested Project Topics (cont.)

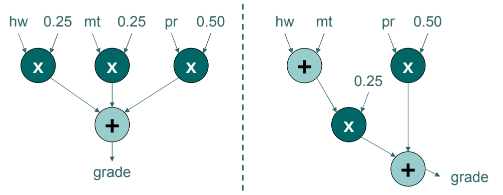
- Prototype some microarchitectural concept using FPGA
  - See proceedings of MICRO/ISCA/HPCA/ASPLOS from last 5 years
  - Survey some recurring topic
  - Compare results from simulation (Simplescalar) to FPGA prototype results
- Evaluation of various high-level synthesis tools and methodology
  - Survey 4-5 different open-source high-level synthesis tools
  - Pick a representative (pre-existing) benchmark set, see how they fare…how well do they work?
  - Compare to microprocessor-based implementation of same
- Others from past years (see CprE 583 web page)
- Anything else that interests you!

Recap

- **Reconfigurable Computing:**
  - (1) systems incorporating some form of hardware programmability – customizing how the hardware is used using a number of physical control points [Compton, 2002]
  - (2) computing via a post-fabrication and spatially programmed connection of processing elements [Wawrzynek, 2004]
  - (3) general-purpose custom hardware [Goldstein, 1998]

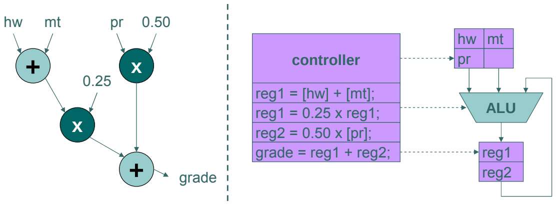
Spatial Mapping

grade = 0.25*homework + 0.25*midterm + 0.50*project



- A hardware resource (multiplier or adder) is allocated for each operator in the compute graph
- The compute graph is transformed directly into the implementation template

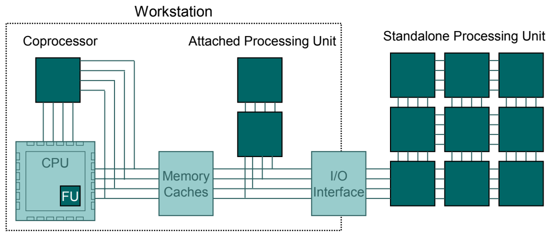
Temporal Mapping



```
reg1 = [hw] + [mt];
reg1 = 0.25 x reg1;
reg2 = 0.50 x [pr];
grade = reg1 + reg2;
```

- A hardware resource (ALU) is time-multiplexed to implement the actions of the operators in the compute graph
- Sequential / general purpose / software solution

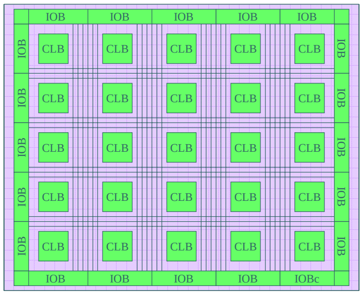
Coupling in a Reconfigurable System



- Some advantages of each?
- Some disadvantages?

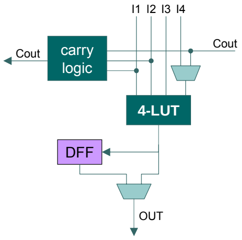
Generic FPGA Architecture

- **FPGA = F**ield-**P**rogrammable **G**ate **A**rray
  - Input/Output Buffers (IOBs)
  - Configurable Logic Blocks (CLBs)
  - Programmable interconnect mesh



Island-style FPGA architecture

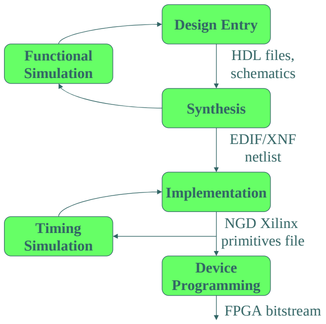
LUT-based Logic Element



- Each LUT operates on four one-bit inputs
- Output is one data bit
- Can perform any Boolean function of four inputs
- 2^{2^4} = 65536 functions (4096 patterns)

- The basic logic element can be more complex
- *Coarse* v. *Fine* grained
- Contains some sort of programmable interconnect

Sample FPGA Design Flow (Xilinx)



FPGAs are **RE**configurable…

- …but good luck getting it to work!
  - Commercial tool support is nonexistent
  - Not ready for prime-time
- Uses for reconfiguration
  - Product life extension
  - Tolerance for manufacturing faults
- **Runtime** reconfiguration – time multiplexing specialized circuits to make more efficient use of existing resources
- **Dynamic** reconfiguration – hardware sharing (multiplexing) on the fly
  - DPGA v. FPGA
  - Active area of research

Outline

- Recap
- Design Exercise: The Quadratic Equation
- Terms and Definitions
- Measuring Computing Density
- Quantitatively Comparing Computing Machines

Design Exercise

- Consider the function: $y = Ax^2 + Bx + C$
- In groups of 2, design an architecture for this function
  - Building blocks – adders, multipliers, muxes
  - Don't worry too much about control or timing
- Best circuit design wins a prize

Various Possible Designs

Design A

Design B

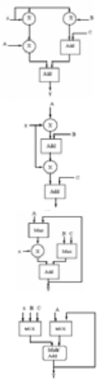
Design C

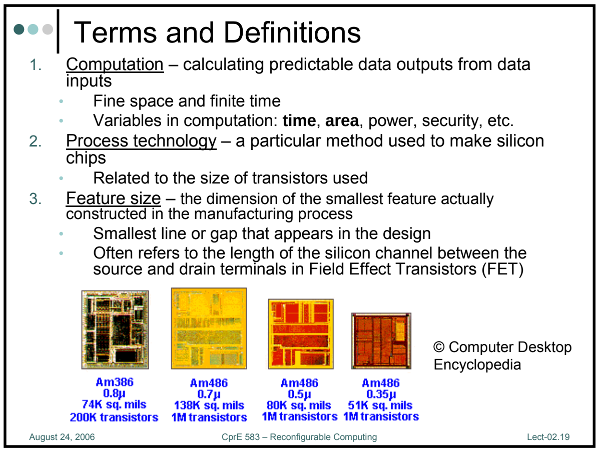
Design D

•Which one is t

Comparing Different Designs

- Design A
  - Requires 3 multiply and 2 add area units
  - Requires 2 multiply and 1 add time units
- Design B
  - Requires 2 multiply and 2 add area units
  - Requires 2 multiply and 2 add time units
- Design C
  - Requires 1 multiply, 1 add, and 2 2:1 mux area units
  - Requires 2 multiply and 2 add time units
- Design D
  - Requires 1 compound add/multiply unit, 1 3:1 mux, and 1 2:1 mux area units
  - Requires 2 multiply and 2 add time units





Terms and Definitions

1. Computation – calculating predictable data outputs from data inputs
   - Fine space and finite time
   - Variables in computation: **time**, **area**, power, security, etc.
2. Process technology – a particular method used to make silicon chips
   - Related to the size of transistors used
3. Feature size – the dimension of the smallest feature actually constructed in the manufacturing process
   - Smallest line or gap that appears in the design
   - Often refers to the length of the silicon channel between the source and drain terminals in Field Effect Transistors (FET)

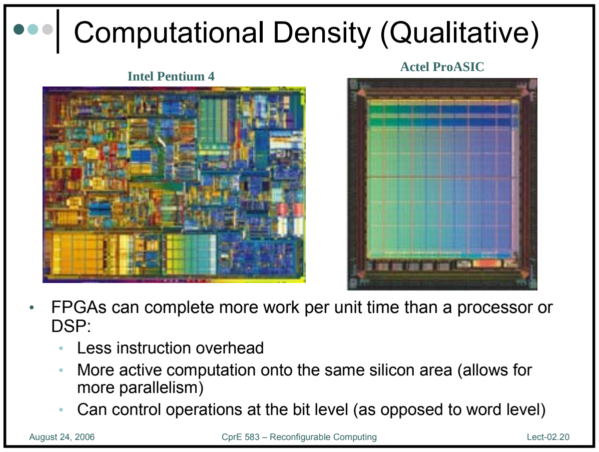
Am386
0.8µ
74K sq. mils
200K transistors

Am486
0.7µ
138K sq. mils
1M transistors

Am486
0.5µ
80K sq. mils
1M transistors

Am486
0.35µ
51K sq. mils
1M transistors

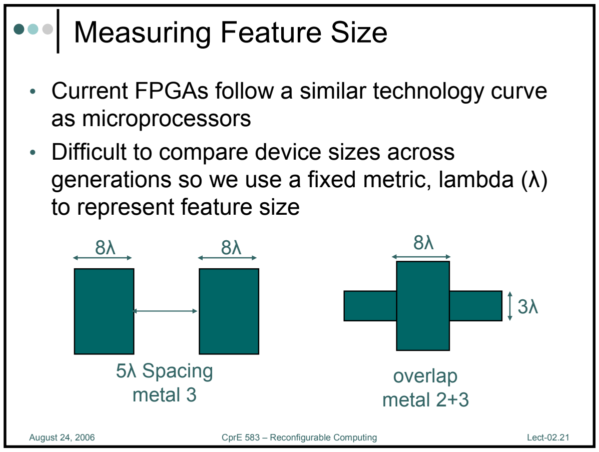
© Computer Desktop Encyclopedia



Computational Density (Qualitative)

Intel Pentium 4
Actel ProASIC

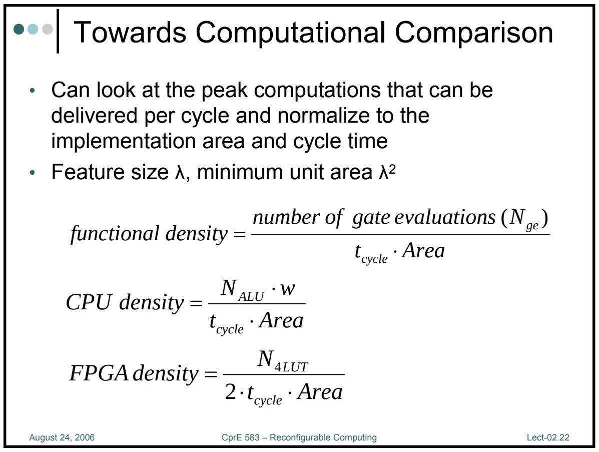
- FPGAs can complete more work per unit time than a processor or DSP:
  - Less instruction overhead
  - More active computation onto the same silicon area (allows for more parallelism)
  - Can control operations at the bit level (as opposed to word level)



Measuring Feature Size

- Current FPGAs follow a similar technology curve as microprocessors
- Difficult to compare device sizes across generations so we use a fixed metric, lambda (λ) to represent feature size

8λ
8λ
5λ Spacing
metal 3
8λ
3λ
overlap
metal 2+3



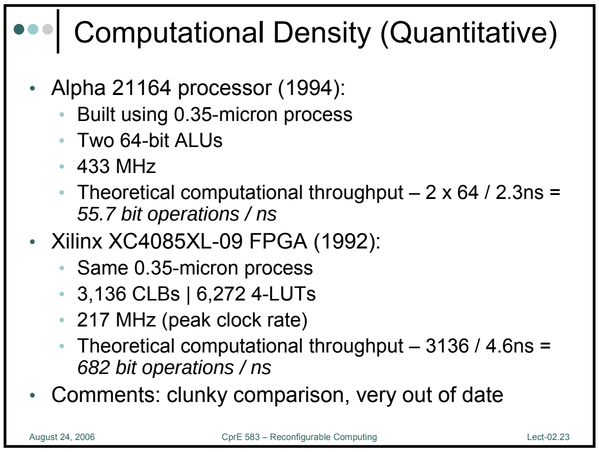
Towards Computational Comparison

- Can look at the peak computations that can be delivered per cycle and normalize to the implementation area and cycle time
- Feature size λ, minimum unit area λ²

$$functional\ density = \frac{number\ of\ \ gate\ evaluations\ (N_{ge})}{t_{cycle} \cdot Area}$$

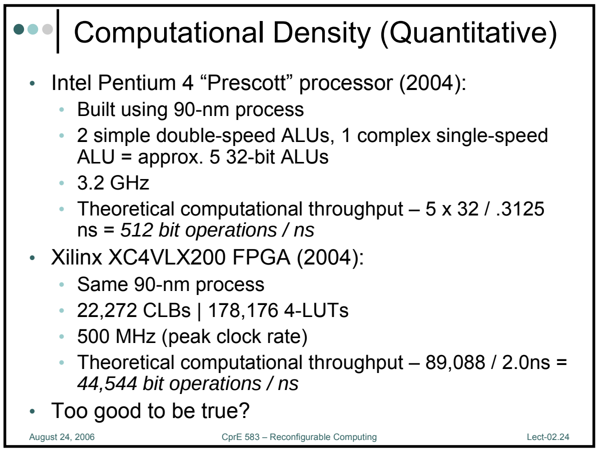
$$CPU\ density = \frac{N_{ALU} \cdot w}{t_{cycle} \cdot Area}$$

$$FPGA\ density = \frac{N_{4LUT}}{2 \cdot t_{cycle} \cdot Area}$$



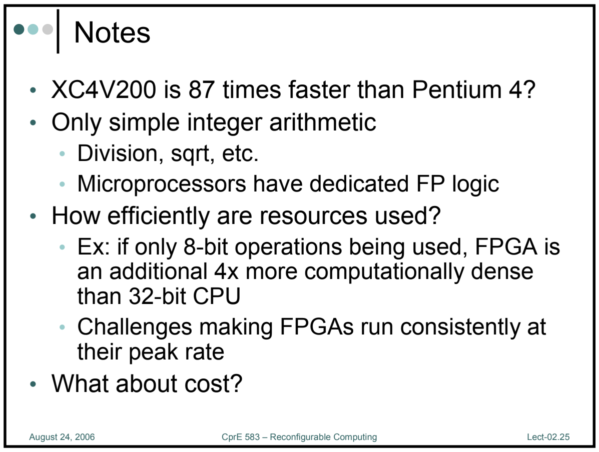
Computational Density (Quantitative)

- Alpha 21164 processor (1994):
  - Built using 0.35-micron process
  - Two 64-bit ALUs
  - 433 MHz
  - Theoretical computational throughput – 2 x 64 / 2.3ns = *55.7 bit operations / ns*
- Xilinx XC4085XL-09 FPGA (1992):
  - Same 0.35-micron process
  - 3,136 CLBs | 6,272 4-LUTs
  - 217 MHz (peak clock rate)
  - Theoretical computational throughput – 3136 / 4.6ns = *682 bit operations / ns*
- Comments: clunky comparison, very out of date



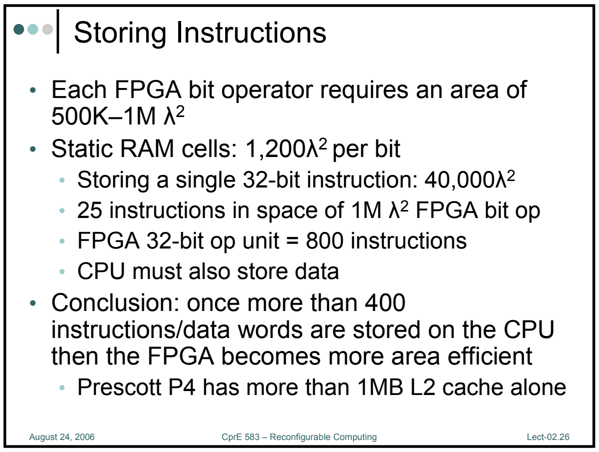
Computational Density (Quantitative)

- Intel Pentium 4 “Prescott” processor (2004):
  - Built using 90-nm process
  - 2 simple double-speed ALUs, 1 complex single-speed ALU = approx. 5 32-bit ALUs
  - 3.2 GHz
  - Theoretical computational throughput – 5 x 32 / .3125 ns = *512 bit operations / ns*
- Xilinx XC4VLX200 FPGA (2004):
  - Same 90-nm process
  - 22,272 CLBs | 178,176 4-LUTs
  - 500 MHz (peak clock rate)
  - Theoretical computational throughput – 89,088 / 2.0ns = *44,544 bit operations / ns*
- Too good to be true?



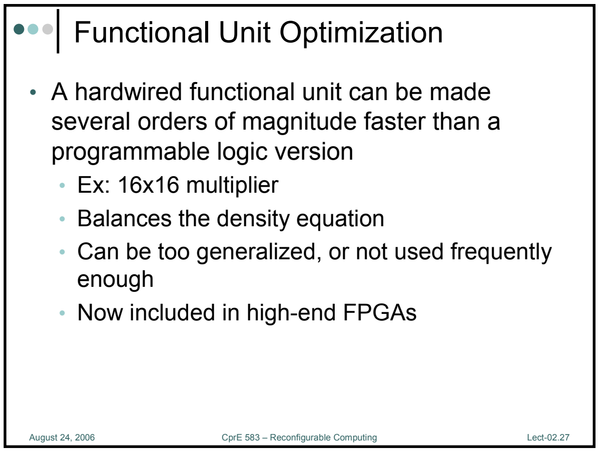
Notes

- XC4V200 is 87 times faster than Pentium 4?
- Only simple integer arithmetic
  - Division, sqrt, etc.
  - Microprocessors have dedicated FP logic
- How efficiently are resources used?
  - Ex: if only 8-bit operations being used, FPGA is an additional 4x more computationally dense than 32-bit CPU
  - Challenges making FPGAs run consistently at their peak rate
- What about cost?



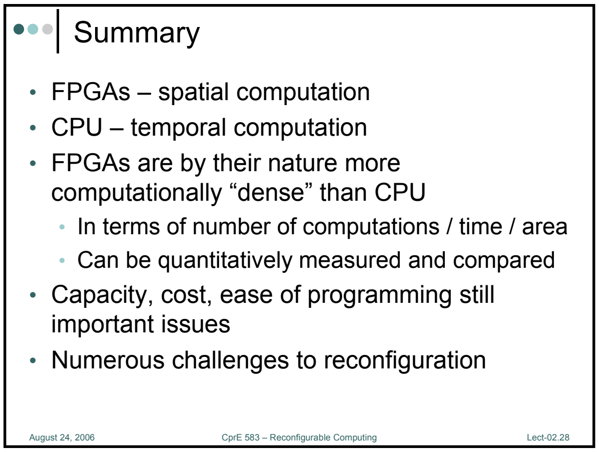
Storing Instructions

- Each FPGA bit operator requires an area of 500K–1M λ^2
- Static RAM cells: 1,200λ^2 per bit
  - Storing a single 32-bit instruction: 40,000λ^2
  - 25 instructions in space of 1M λ^2 FPGA bit op
  - FPGA 32-bit op unit = 800 instructions
  - CPU must also store data
- Conclusion: once more than 400 instructions/data words are stored on the CPU then the FPGA becomes more area efficient
  - Prescott P4 has more than 1MB L2 cache alone



Functional Unit Optimization

- A hardwired functional unit can be made several orders of magnitude faster than a programmable logic version
  - Ex: 16x16 multiplier
  - Balances the density equation
  - Can be too generalized, or not used frequently enough
  - Now included in high-end FPGAs



Summary

- FPGAs – spatial computation
- CPU – temporal computation
- FPGAs are by their nature more computationally “dense” than CPU
  - In terms of number of computations / time / area
  - Can be quantitatively measured and compared
- Capacity, cost, ease of programming still important issues
- Numerous challenges to reconfiguration