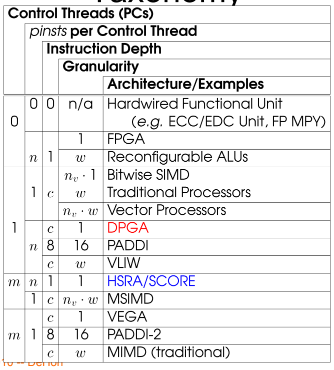
ESE534: Computer Organization

Day 11: March 1, 2010
Instruction Space Modeling

Last Time

- Instruction Requirements
- Instruction Space

Architecture Instruction Taxonomy



Control Threads (PCs)	*pinsts* per Control Thread	Instruction Depth	Granularity	Architecture/Examples
0	0	0	n/a	Hardwired Functional Unit (*e.g.* ECC/EDC Unit, FP MPY)
	n	1	1	FPGA
			w	Reconfigurable ALUs
1	1	c	$n_v \cdot 1$	Bitwise SIMD
			w	Traditional Processors
			$n_v \cdot w$	Vector Processors
	n	c	1	DPGA
		8	16	PADDI
		c	w	VLIW
m	n	1	1	HSRA/SCORE
	1	c	$n_v \cdot w$	MSIMD
m	1	c	1	VEGA
		8	16	PADDI-2
		c	w	MIMD (traditional)

Architecture Taxonomy

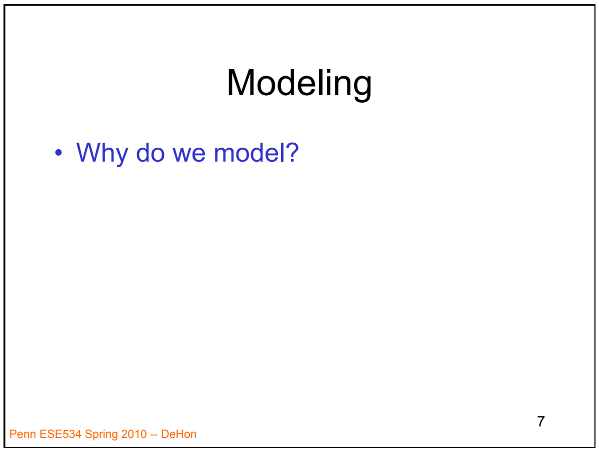
PCs	Pints/PC	depth	width	Architecture
0	1	1	1	FPGA
1	1	1024	32	Scalar Processor (RISC)
1	N	D	W	VLIW (superscalar)
1	1	Small	W*N	SIMD, GPU, Vector
N	1	D	W	MIMD
4	4	2048	64	Quad core

Today

- Model Architecture from Instruction Parameters
  – implied costs
  – gross application characteristics

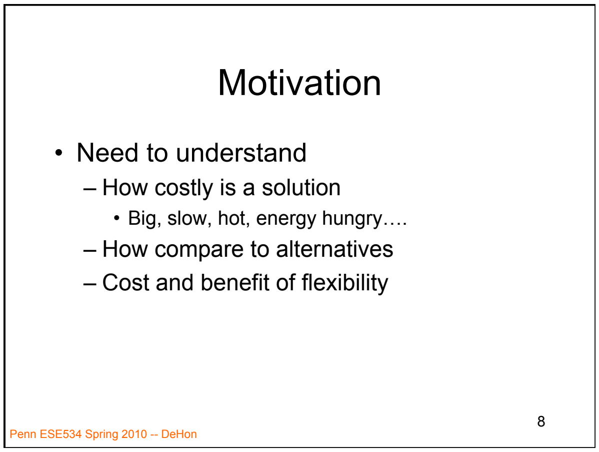
Quotes

- *If it can't be expressed in figures, it is not science; it is opinion.* -- Lazarus Long



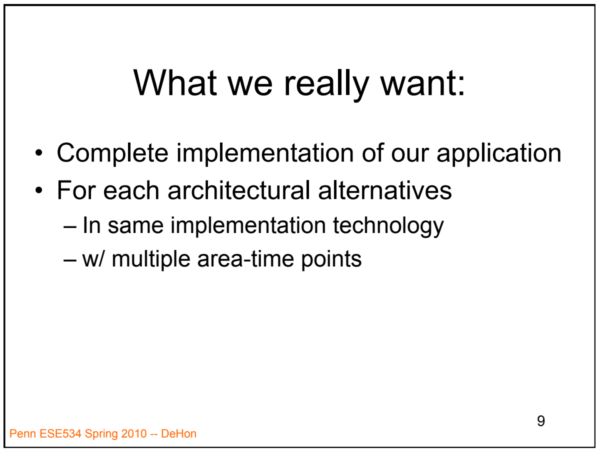
Modeling

- Why do we model?



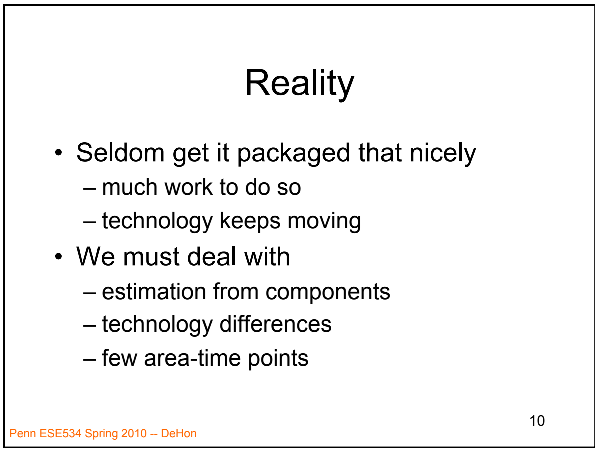
Motivation

- Need to understand
  - How costly is a solution
    - Big, slow, hot, energy hungry….
  - How compare to alternatives
  - Cost and benefit of flexibility



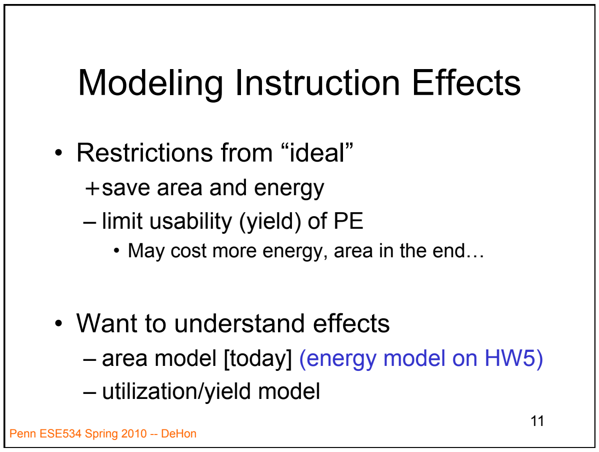
What we really want:

- Complete implementation of our application
- For each architectural alternatives
  - In same implementation technology
  - w/ multiple area-time points



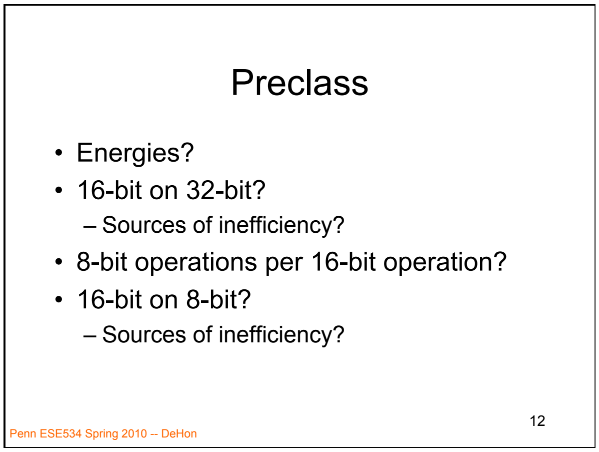
Reality

- Seldom get it packaged that nicely
  - much work to do so
  - technology keeps moving
- We must deal with
  - estimation from components
  - technology differences
  - few area-time points



Modeling Instruction Effects

- Restrictions from “ideal”
  +save area and energy
  - limit usability (yield) of PE
    - May cost more energy, area in the end…
- Want to understand effects
  - area model [today] (energy model on HW5)
  - utilization/yield model

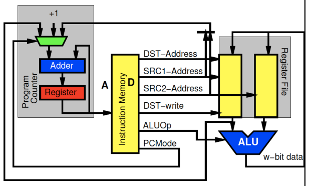


Preclass

- Energies?
- 16-bit on 32-bit?
  - Sources of inefficiency?
- 8-bit operations per 16-bit operation?
- 16-bit on 8-bit?
  - Sources of inefficiency?

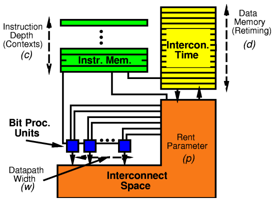
Efficiency/Yield Intuition

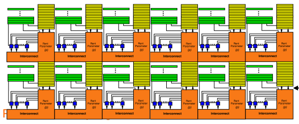
- What happens when
  - Datapath is too wide?
  - Datapath is too narrow?
  - Instruction memory is too deep?
  - Instruction memory is too shallow?



Computing Device

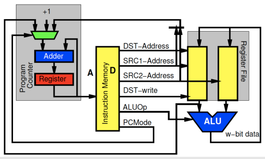
- Composition
  - Bit Processing elements
  - Interconnect: space
  - Interconnect: time
  - Instruction Memory

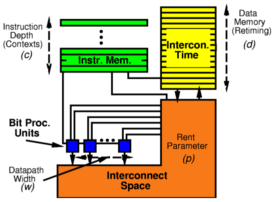


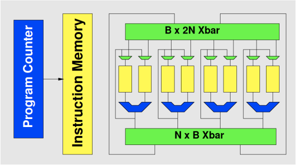


Tile together to build device

Computing Device

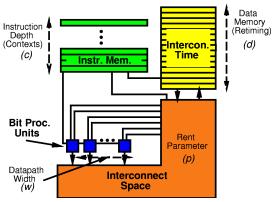


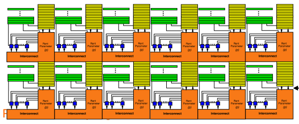




Computing Device

- Composition
  - Bit Processing elements
  - Interconnect: space
  - Interconnect: time
  - Instruction Memory

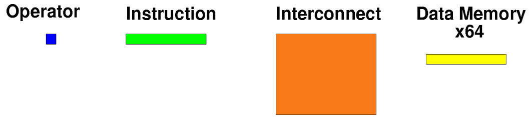




Tile together to build device

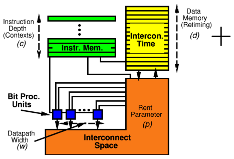
Relative Sizes

• Bit Operator	10-20Kλ^2
• Bit Operator Interconnect	500K-1Mλ^2
• Instruction (w/ interconnect)	80Kλ^2
• Memory bit (SRAM)	1-2Kλ^2

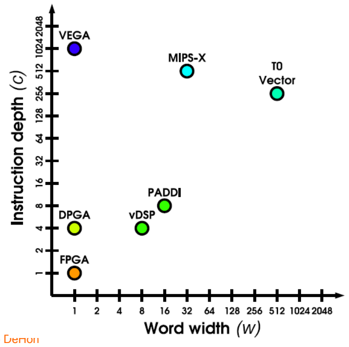


Model Area

$$A_{bit\_elm} = A_{fixed} + \underbrace{N_{SW}(N_p, w, p) \cdot A_{SW}}_{\textbf{interconnect}} + \underbrace{\left(\frac{c}{w}\right) \cdot n_{ibits} \cdot A_{mem\_cell}}_{\textbf{instruction memory}} + \underbrace{d \cdot A_{mem\_cell}}_{\textbf{retiming memory}}$$



Architectures Fall in Space

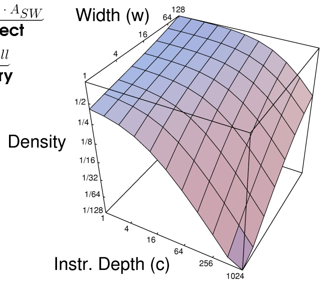


Calibrate Model

FPGA	**model** $w = 1$, $d = c = 1$, $k = 4$	**880K**λ^2
	Xilinx 4K	**630K**λ^2
	Altera 8K	**930K**λ^2
SIMD	**model** $w = 1000$, $c = 0$, $d = 64$, $k = 3$	**170K**λ^2
	Abacus	**190K**λ^2
Processor	**model** $w = 32$, $d = 32$, $c = 1024$, $k = 2$	**2.6M**λ^2
	MIPS-X	**2.1M**λ^2

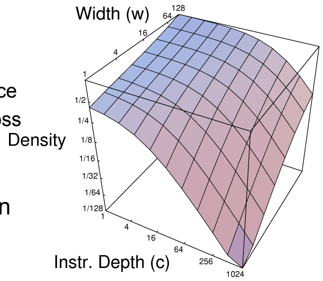
Peak Densities from Model

$$A_{bit\_elm} = A_{fixed} + \underbrace{N_{SW}(N_p, w, p) \cdot A_{SW}}_{\textbf{interconnect}} + \underbrace{\left(\frac{c}{w}\right) \cdot n_{ibits} \cdot A_{mem\_cell}}_{\textbf{instruction memory}} + \underbrace{d \cdot A_{mem\_cell}}_{\textbf{retiming memory}}$$

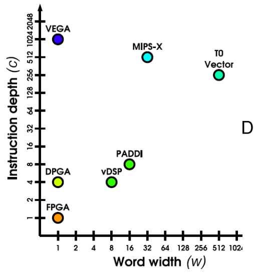


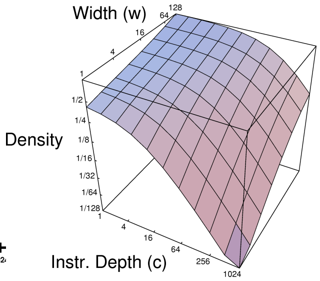
Peak Densities from Model

- Only 2 of 4 parameters
  - small slice of space
  - 100× density across
- Large difference in peak densities
  - large design space!



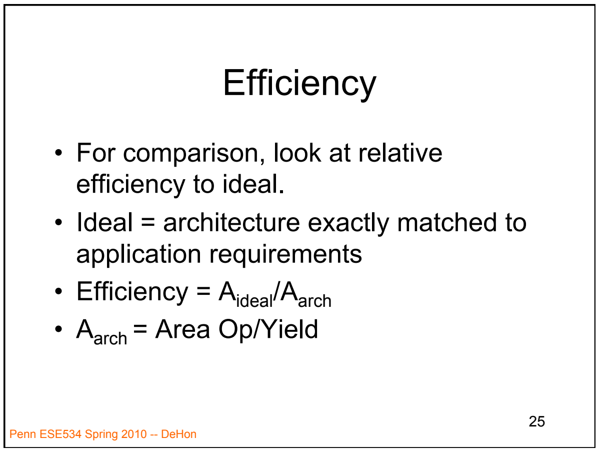
Architectural parameters → Peak Densities





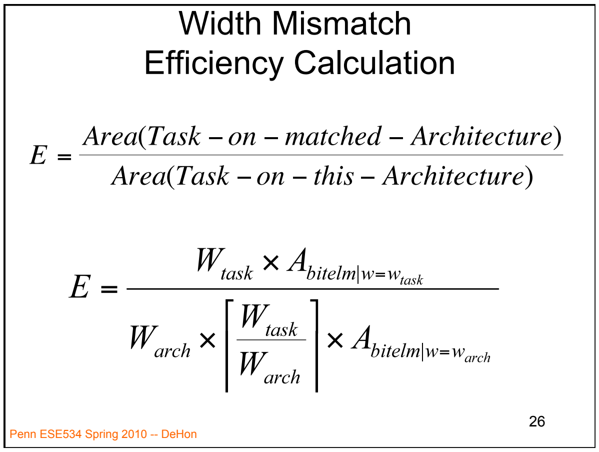
Efficiency

- What do we really want to maximize?
  - Not peak, “guaranteed not to exceed” performance, but…
  - Useful work per unit silicon [per Joule]
- Yield Fraction / Area
- (or minimize (Area/Yielded performance))



Efficiency

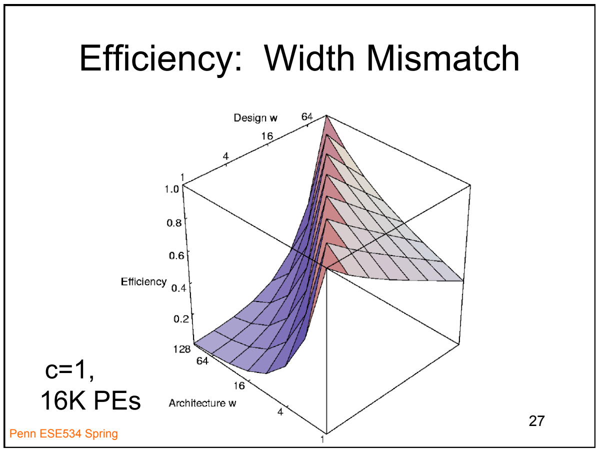
- For comparison, look at relative efficiency to ideal.
- Ideal = architecture exactly matched to application requirements
- Efficiency = A_{ideal}/A_{arch}
- A_{arch} = Area Op/Yield



Width Mismatch Efficiency Calculation

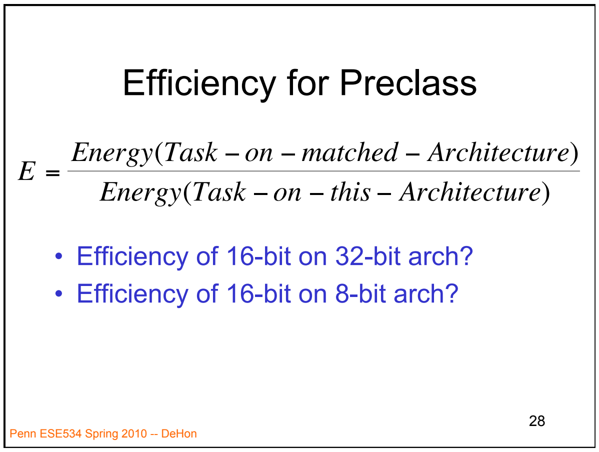
$$E = \frac{Area(Task-on-matched-Architecture)}{Area(Task-on-this-Architecture)}$$

$$E = \frac{W_{task} \times A_{bitelm|w=w_{task}}}{W_{arch} \times \left\lceil \frac{W_{task}}{W_{arch}} \right\rceil \times A_{bitelm|w=w_{arch}}}$$



Efficiency: Width Mismatch

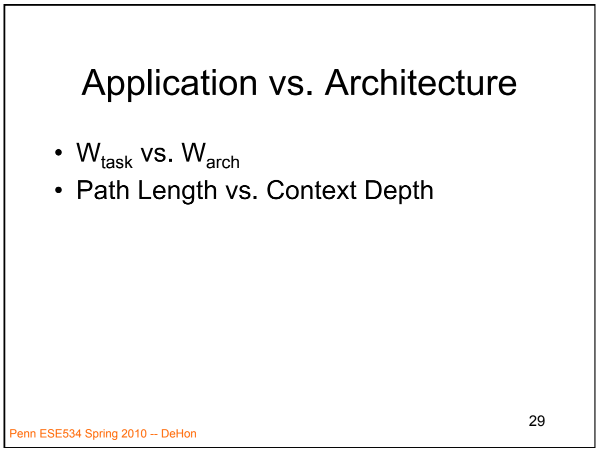
c=1,
16K PEs



Efficiency for Preclass

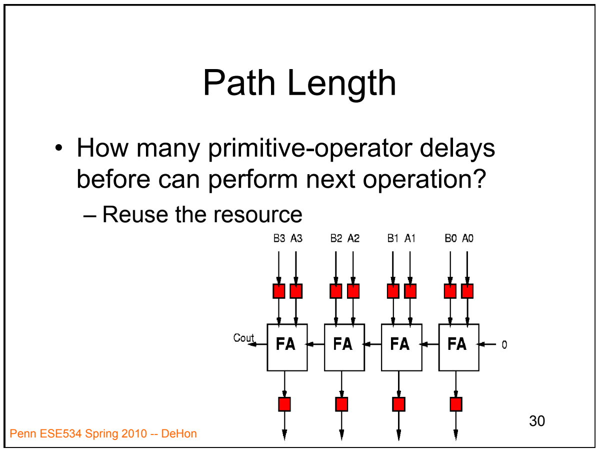
$$E = \frac{Energy(Task-on-matched-Architecture)}{Energy(Task-on-this-Architecture)}$$

- Efficiency of 16-bit on 32-bit arch?
- Efficiency of 16-bit on 8-bit arch?



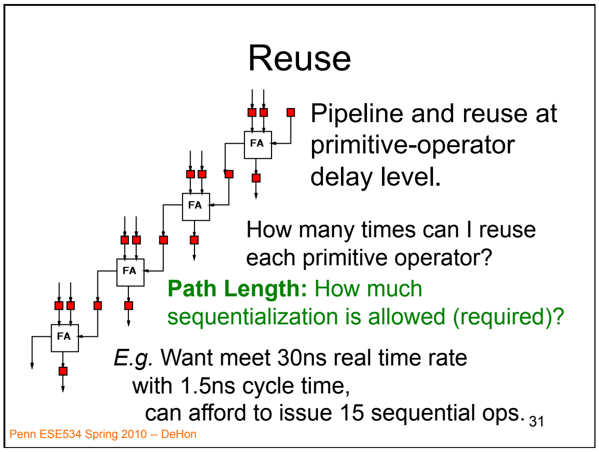
Application vs. Architecture

- W_{task} vs. W_{arch}
- Path Length vs. Context Depth



Path Length

- How many primitive-operator delays before can perform next operation?
  - Reuse the resource



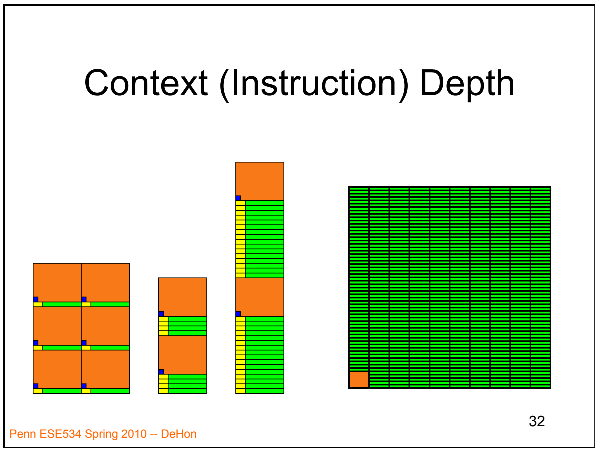
Reuse

- Pipeline and reuse at primitive-operator delay level.

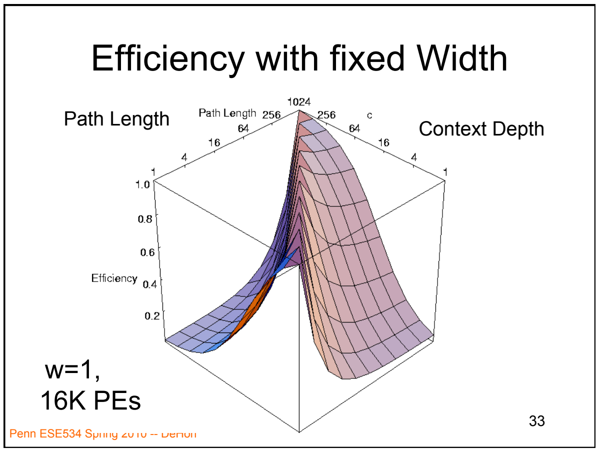
How many times can I reuse each primitive operator?

Path Length: How much sequentialization is allowed (required)?

E.g. Want meet 30ns real time rate with 1.5ns cycle time, can afford to issue 15 sequential ops.



Context (Instruction) Depth

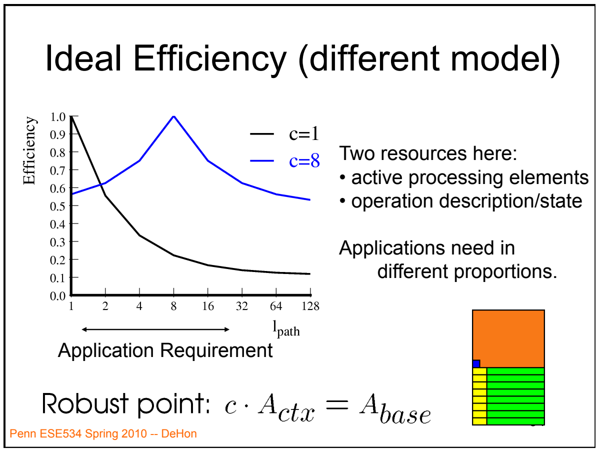


Efficiency with fixed Width

Path Length

Context Depth

w=1,
16K PEs



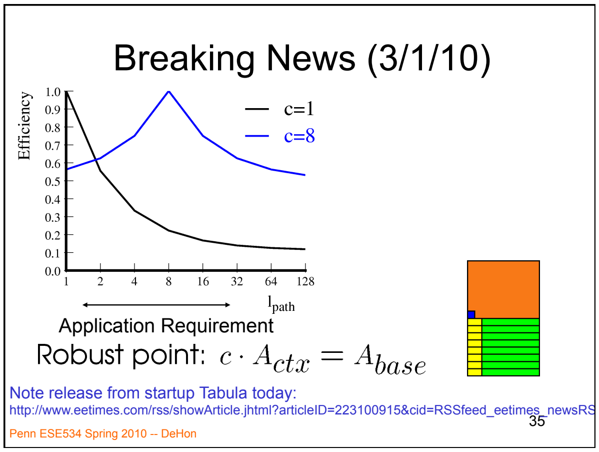
Ideal Efficiency (different model)

Two resources here:
- active processing elements
- operation description/state

Applications need in different proportions.

Application Requirement

Robust point: $c \cdot A_{ctx} = A_{base}$

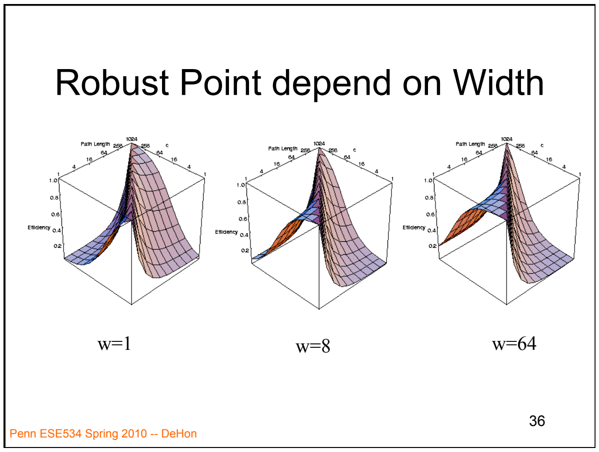


Breaking News (3/1/10)

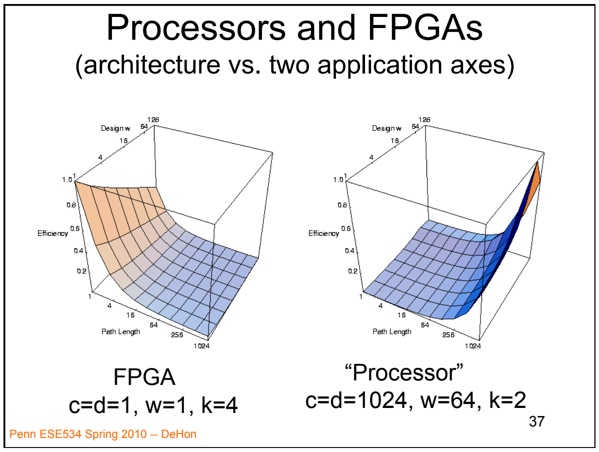
Application Requirement

Robust point: $c \cdot A_{ctx} = A_{base}$

Note release from startup Tabula today:
http://www.eetimes.com/rss/showArticle.jhtml?articleID=223100915&cid=RSSfeed_eetimes_newsRS



Robust Point depend on Width

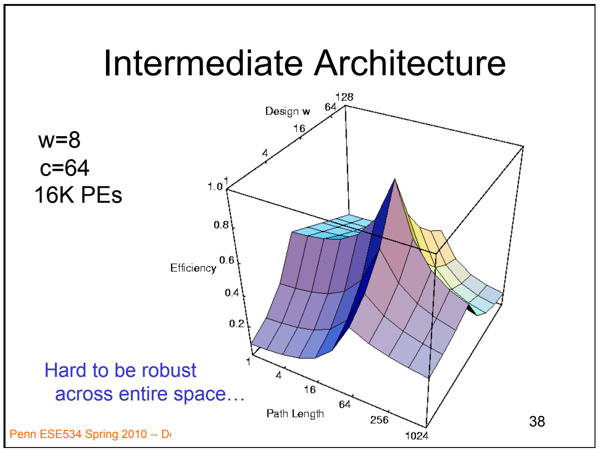


Processors and FPGAs

(architecture vs. two application axes)

FPGA
c=d=1, w=1, k=4

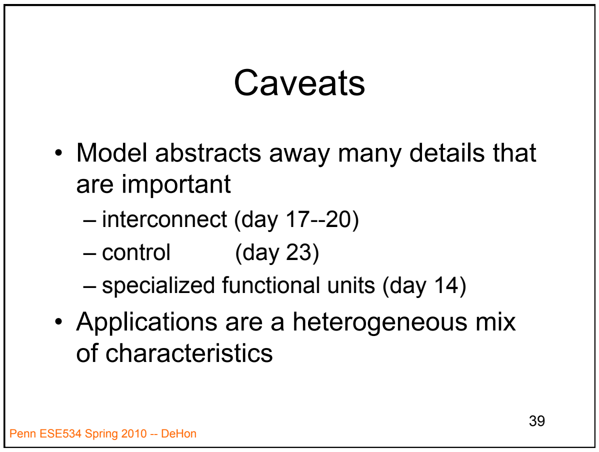
“Processor”
c=d=1024, w=64, k=2



Intermediate Architecture

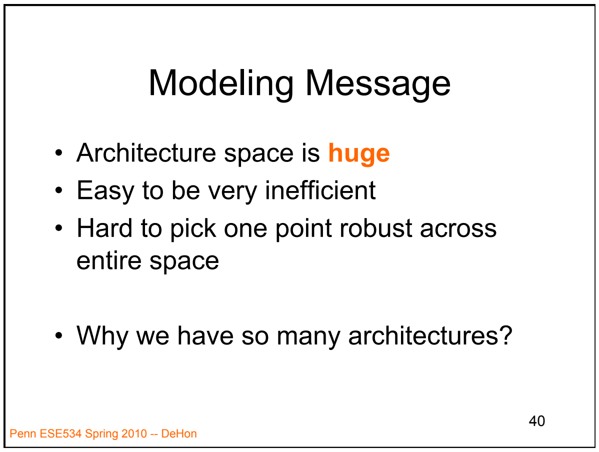
w=8
c=64
16K PEs

Hard to be robust
across entire space…



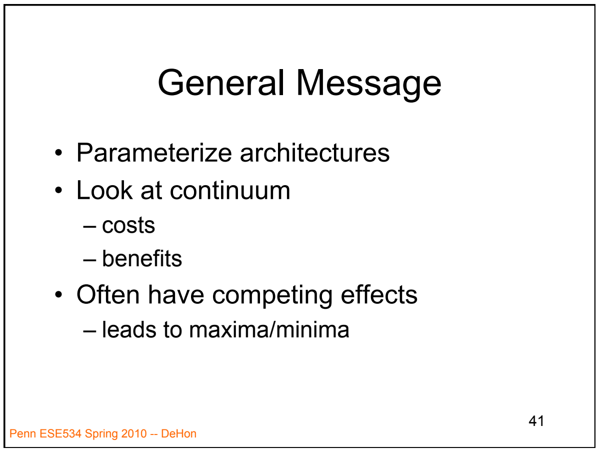
Caveats

- Model abstracts away many details that are important
  - interconnect (day 17--20)
  - control (day 23)
  - specialized functional units (day 14)
- Applications are a heterogeneous mix of characteristics



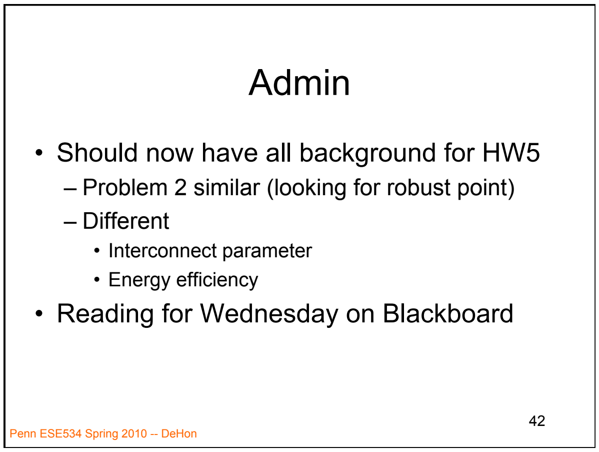
Modeling Message

- Architecture space is **huge**
- Easy to be very inefficient
- Hard to pick one point robust across entire space
- Why we have so many architectures?



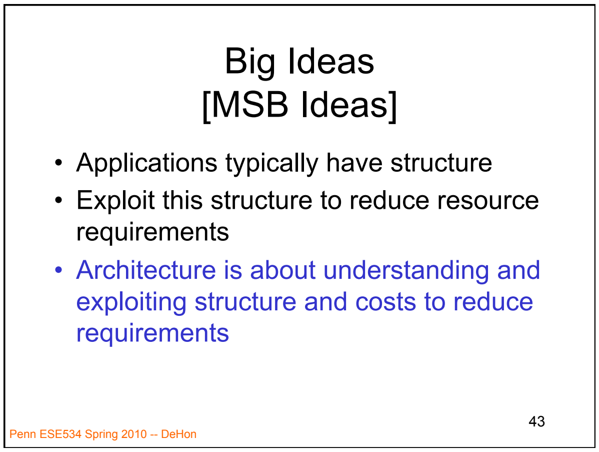
General Message

- Parameterize architectures
- Look at continuum
  - costs
  - benefits
- Often have competing effects
  - leads to maxima/minima



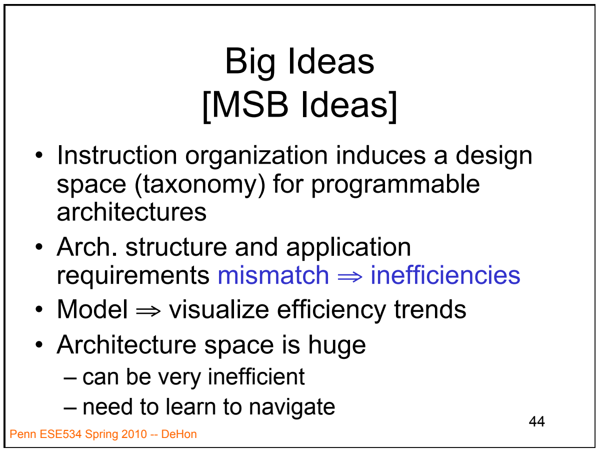
Admin

- Should now have all background for HW5
  - Problem 2 similar (looking for robust point)
  - Different
    - Interconnect parameter
    - Energy efficiency
- Reading for Wednesday on Blackboard



Big Ideas [MSB Ideas]

- Applications typically have structure
- Exploit this structure to reduce resource requirements
- Architecture is about understanding and exploiting structure and costs to reduce requirements



Big Ideas [MSB Ideas]

- Instruction organization induces a design space (taxonomy) for programmable architectures
- Arch. structure and application requirements mismatch ⇒ inefficiencies
- Model ⇒ visualize efficiency trends
- Architecture space is huge
  - can be very inefficient
  - need to learn to navigate