CS184a: Computer Architecture (Structure and Organization)

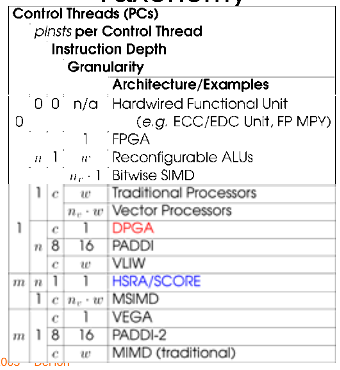
Day 9: January 26, 2005

Modeling Instruction Space and Empirical Comparisons

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
			$n_v \cdot 1$	Bitwise SIMD
1	1	c	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)

Today

- Instructions
  - Model Architecture
    - implied costs
    - gross application characteristics
- Empirical Data
  - Processors
  - FPGAs
  - Custom
    - Gate Array
    - Std. Cell
    - Full

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 (big) is a solution
  - – 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
- Deal with
  - – estimation from components
  - – technology differences
  - – few area-time points

Modeling Instruction Effects

- Restrictions from “ideal” save area
- Restriction from “ideal” limits usability (yield) of PE

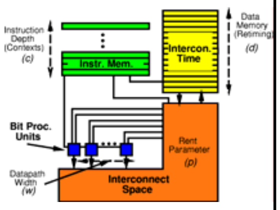
- Want to understand effects
  - – area model
  - – utilization/yield model

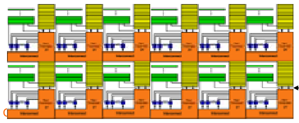
Efficiency/Yield Intuition

- 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

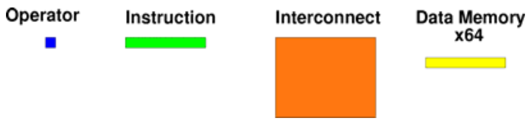




Tile together to build device

Relative Sizes

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

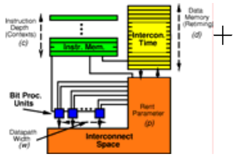


Model Area

$$A_{bit\_elm} = A_{fixed} + \underbrace{N_{SW}(N_p, w, p) \cdot A_{SW}}_{\textbf{interconnect}}$$

$$+ \underbrace{\left(\frac{c}{w}\right) \cdot n_{ibits} \cdot A_{mem\_cell}}_{\textbf{instruction memory}}$$

$$+ \underbrace{d \cdot A_{mem\_cell}}_{\textbf{retiming memory}}$$

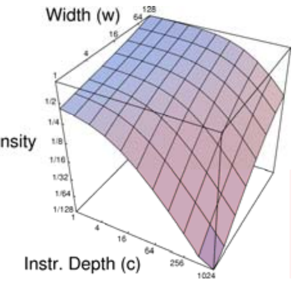


Calibrate Model

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

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



Efficiency

- What do we want to maximize?
  - Useful work per unit silicon
  - (not potential/peak work)
- Yield Fraction / Area
- (or minimize (Area/Yield))

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

Efficiency Calculation

$$\textbf{Efficiency} = \frac{A_{matched\_arch}}{A_{arch}}$$

E.g.

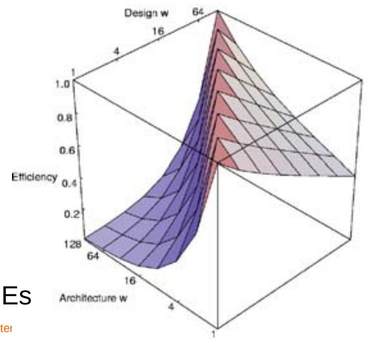
If $w_{task} > w_{arch}$:

$$\textbf{Efficiency} = \frac{w_{task} \times A_{bit\_elm|w=w_{task}}}{\left\lceil \frac{w_{task}}{w_{arch}} \right\rceil \times w_{arch} \times A_{bit\_elm|w=w_{arch}}}$$

If $w_{task} < w_{arch}$:

$$\textbf{Efficiency} = \frac{w_{task} \times A_{bit\_elm|w=w_{task}}}{w_{arch} \times A_{bit\_elm|w=w_{arch}}}$$

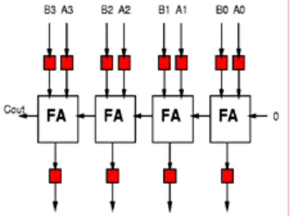
Efficiency: Width Mismatch



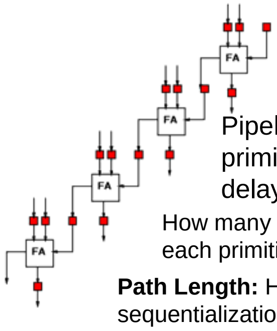
c=1,
16K PEs

Path Length

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



Reuse

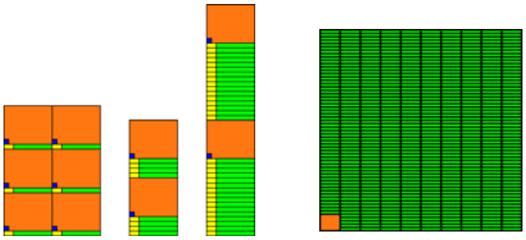


Pipeline and reuse at primitive-operator delay level.

How many times can I reuse each primitive operator?

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

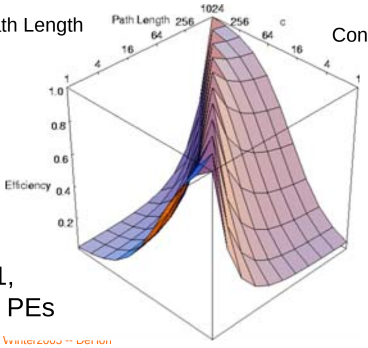
Context Depth



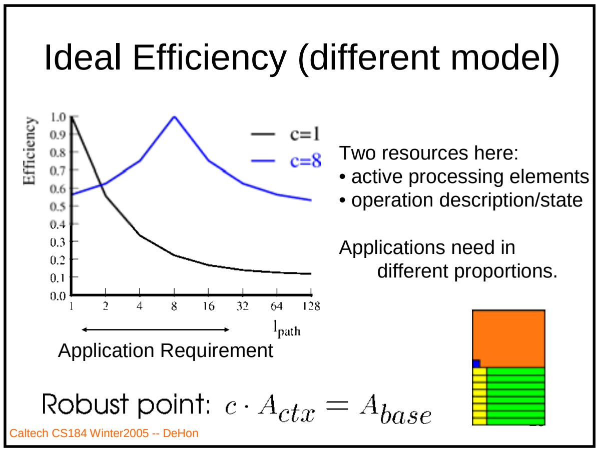
Efficiency with fixed Width

Path Length

Context Depth



w=1,
16K PEs



Ideal Efficiency (different model)

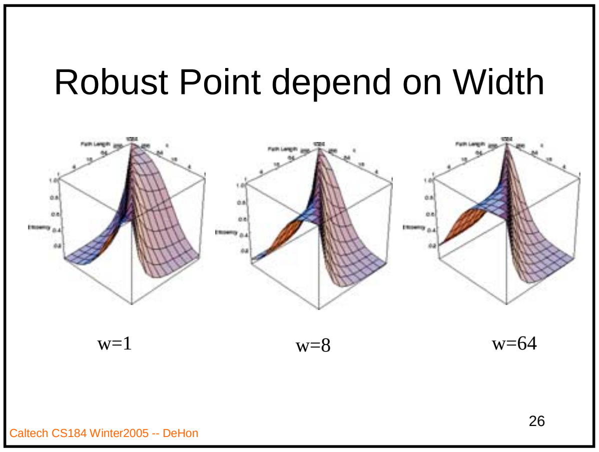
Two resources here:
• active processing elements
• operation description/state

Applications need in different proportions.

Application Requirement

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

Caltech CS184 Winter2005 -- DeHon



Robust Point depend on Width

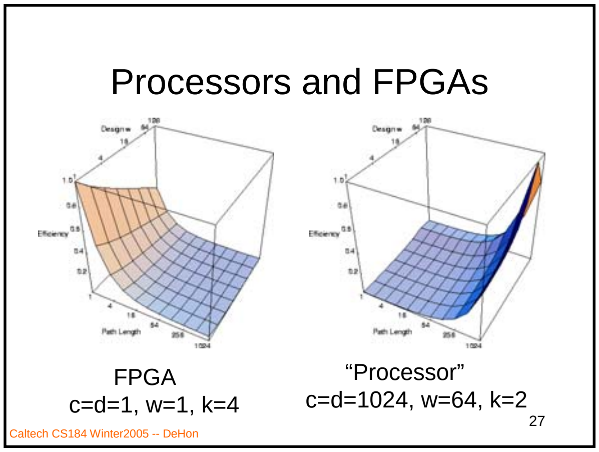
w=1

w=8

w=64

Caltech CS184 Winter2005 -- DeHon

26



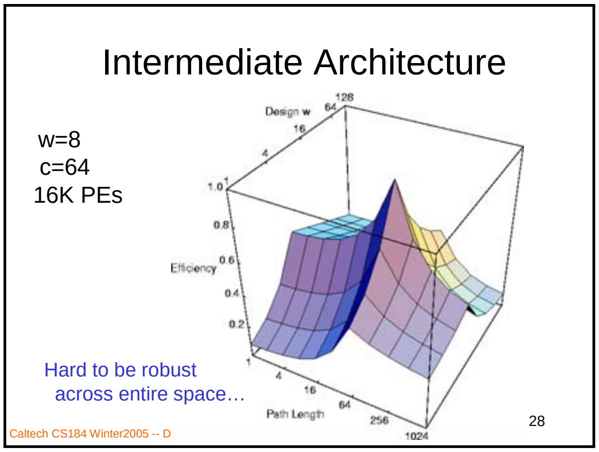
Processors and FPGAs

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

"Processor"
c=d=1024, w=64, k=2

Caltech CS184 Winter2005 -- DeHon

27



Intermediate Architecture

w=8
c=64
16K PEs

Hard to be robust across entire space…

Caltech CS184 Winter2005 -- D

28

Caveats

- Model abstracts away many details which are important
  - interconnect (day 12--17)
  - control (day 21)
  - specialized functional units (next time)
- Applications are a heterogeneous mix of characteristics

Caltech CS184 Winter2005 -- DeHon

29

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?

Caltech CS184 Winter2005 -- DeHon

30

General Message

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

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

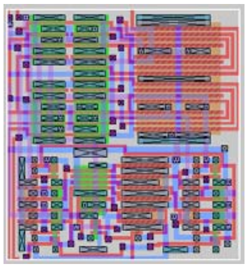
Empirical Comparisons

Empirical

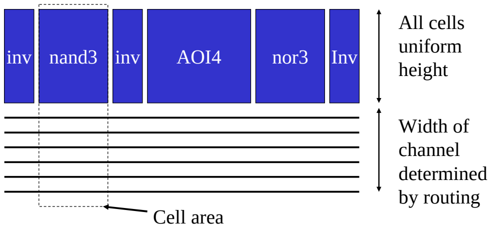
- Ground modeling in some concretes
- Start sorting out
  - custom vs. configurable
  - spatial configurable vs. temporal

Full Custom

- Get to define all layers
- Use any geometry you like
- Only rules are process design rules
- CS181



Standard Cell Area



Identify the full custom and standard cell regions on 386DX die
http://microscope.fsu.edu/chipshots/intel/386dxlarge.html

MPGA

- Metal Programmable Gate Array
- Gates pre-placed (poly, diffusion)
- Only get to define metal connections
  - Cheap – only have to pay for metal mask(s)

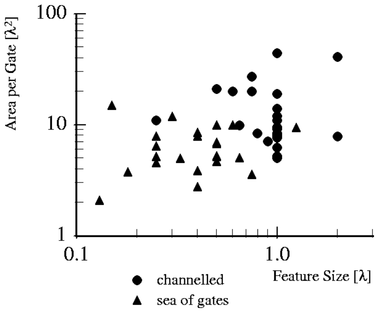
MPGA vs. Custom?

- AMI CICC'83
  - MPGA 1.0
  - Std-Cell 0.7
  - Custom 0.5
- Toshiba DSP
  - Custom 0.3
- Mosaid RAM
  - Custom 0.2

- GE CICC'86
  - MPGA 1.0
  - Std-Cell 0.4--0.7
    - FF/counter 0.7
    - FullAdder 0.4
    - RAM 0.2

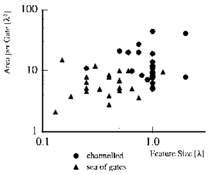
MPGA = Metal Programmable Gate Array
(traditional Gate Array)

Metal Programmable Gate Arrays

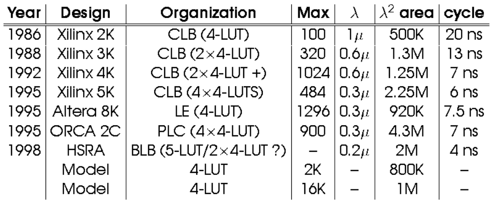


MPGAs

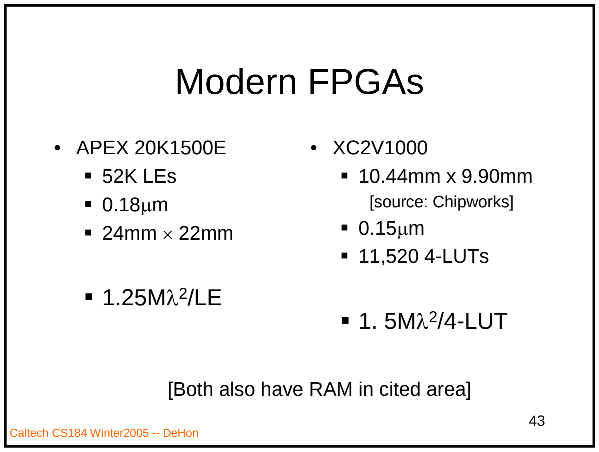
- Modern -- "Sea of Gates"
- yield 35--70%
- maybe 5kλ^2/gate ?
  - (quite a bit of variance)



FPGA Table



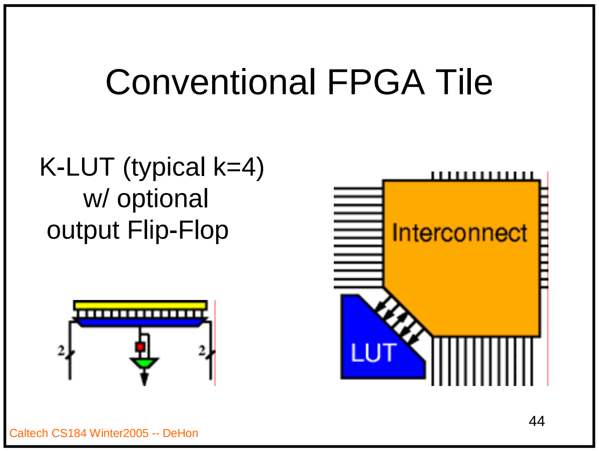
Year	Design	Organization	Max	λ	λ^2 area	cycle
1986	Xilinx 2K	CLB (4-LUT)	100	1μ	500K	20 ns
1988	Xilinx 3K	CLB (2×4-LUT)	320	0.6μ	1.3M	13 ns
1992	Xilinx 4K	CLB (2×4-LUT +)	1024	0.6μ	1.25M	7 ns
1995	Xilinx 5K	CLB (4×4-LUTS)	484	0.3μ	2.25M	6 ns
1995	Altera 8K	LE (4-LUT)	1296	0.3μ	920K	7.5 ns
1995	ORCA 2C	PLC (4×4-LUT)	900	0.3μ	4.3M	7 ns
1998	HSRA	BLB (5-LUT/2×4-LUT ?)	–	0.2μ	2M	4 ns
	Model	4-LUT	2K	–	800K	–
	Model	4-LUT	16K	–	1M	–



Modern FPGAs

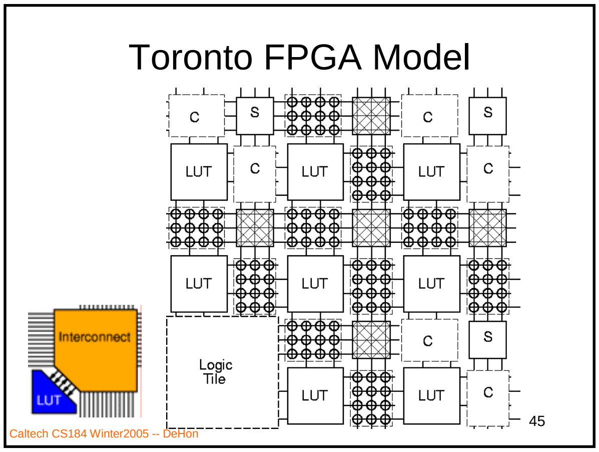
- APEX 20K1500E
  - 52K LEs
  - 0.18μm
  - 24mm × 22mm
  - 1.25Mλ^2/LE
- XC2V1000
  - 10.44mm x 9.90mm
    [source: Chipworks]
  - 0.15μm
  - 11,520 4-LUTs
  - 1. 5Mλ^2/4-LUT

[Both also have RAM in cited area]

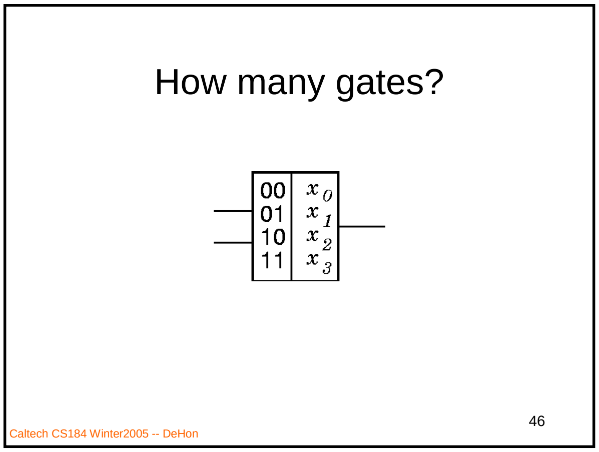


Conventional FPGA Tile

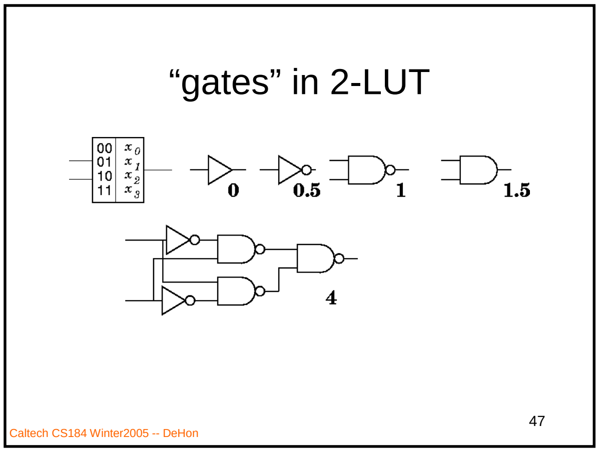
K-LUT (typical k=4)
w/ optional
output Flip-Flop



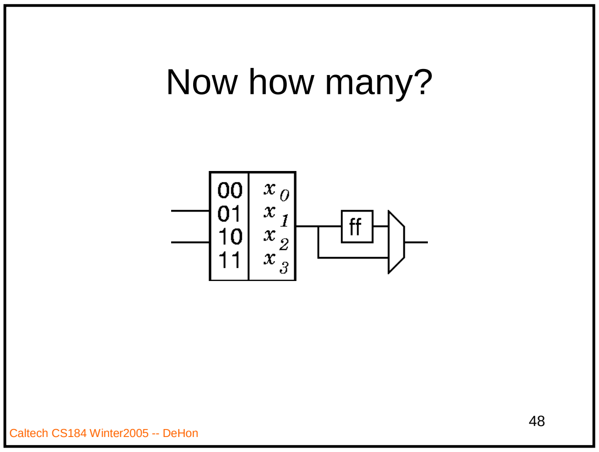
Toronto FPGA Model



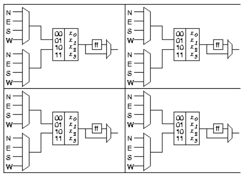
How many gates?

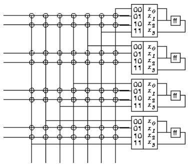


“gates” in 2-LUT



Now how many?



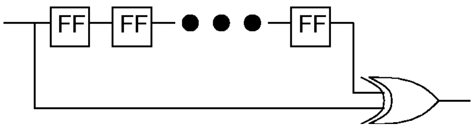


Which gives:

More usable gates?

More gates/unit area?

Gates Required?



Depth=3, Depth=2048?

Gate metric for FPGAs?

- Day8: several components for computations
  - compute element
  - interconnect:
    - space
    - time
  - instructions
- Not all applications need in same **balance**
- Assigning a single “capacity” number to device is an oversimplification

MPGA vs. FPGA

- MPGA (SOG GA)
  - 5Kλ^2/gate
  - 35-70% usable (50%)
  - 7-17Kλ^2/gate net
- Xilinx XC4K
  - 1.25Mλ^2 /CLB
  - 17--48 gates (26?)
  - 26-73Kλ^2/gate net

- Ratio: 2--10 (5)

Adding ~2x Custom/MPGA,
Custom/FPGA ~10x

MPGA vs. FPGA

- MPGA (SOG GA)
  - λ=0.6μ
  - τ_{gd}~1ns
- Xilinx XC4K
  - λ=0.6μ
  - 1–7 gates in 7ns
  - 2-3 gates typical

- Ratio: 1--7 (2.5)

Processors vs. FPGAs

Processors and FPGAs

Metric: $\frac{\textbf{4 input gate-evaluations}}{\lambda^2 \cdot \mathbf{s}}$

Processor: $\frac{2 \times N_{ALU} \times w_{ALU}}{A_{proc} \times t_{cycle}}$ **FPGA:** $\frac{N_{4LUT}}{A_{array} \times t_{cycle}}$

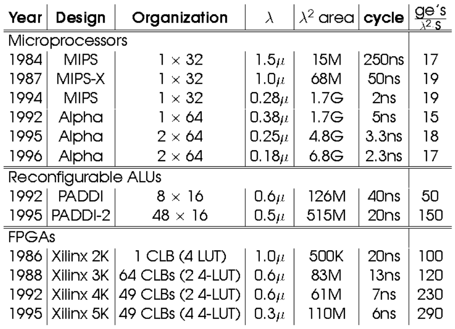
Component Example

- Single die in 0.35µm

XC4085XL-09	3,136 CLBs	4.6ns
	682 Bit Ops/ns	
Alpha 1996	2×64b ALUs	2.3ns
	55.7 Bit Ops/ns	

[1 “bit op” = 2 gate evaluations]

Processors and FPGAs



Year	Design	Organization	λ	λ^2 area	cycle	$\frac{ge's}{\lambda^2 \cdot s}$
Microprocessors						
1984	MIPS	1 × 32	1.5μ	15M	250ns	17
1987	MIPS-X	1 × 32	1.0μ	68M	50ns	19
1994	MIPS	1 × 32	0.28μ	1.7G	2ns	19
1992	Alpha	1 × 64	0.38μ	1.7G	5ns	15
1995	Alpha	2 × 64	0.25μ	4.8G	3.3ns	18
1996	Alpha	2 × 64	0.18μ	6.8G	2.3ns	17
Reconfigurable ALUs						
1992	PADDI	8 × 16	0.6μ	126M	40ns	50
1995	PADDI-2	48 × 16	0.5μ	515M	20ns	150
FPGAs						
1986	Xilinx 2K	1 CLB (4 LUT)	1.0μ	500K	20ns	100
1988	Xilinx 3K	64 CLBs (2 4-LUT)	0.6μ	83M	13ns	120
1992	Xilinx 4K	49 CLBs (2 4-LUT)	0.6μ	61M	7ns	230
1995	Xilinx 5K	49 CLBs (4 4-LUT)	0.3μ	110M	6ns	290

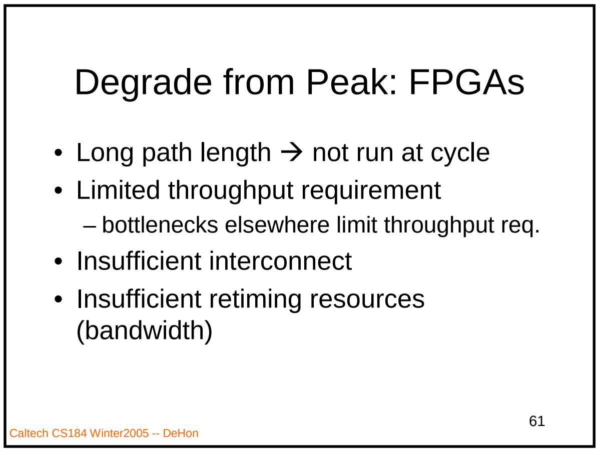
Raw Density Summary

- Area
  - MPGA 2-3x Custom
  - FPGA 5x MPGA
- Area-Time
  - Gate Array 6-10x Custom
  - FPGA 15-20x Gate Array
  - Processor 10x FPGA

Raw Density Caveats

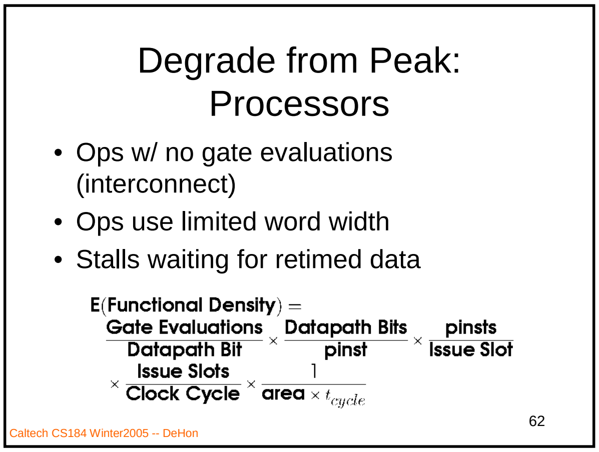
- Processor/FPGA may solve more specialized problem
- Problems have different resource balance requirements
  - …can lead to low yield of raw density

Degrade from Peak



Degrade from Peak: FPGAs

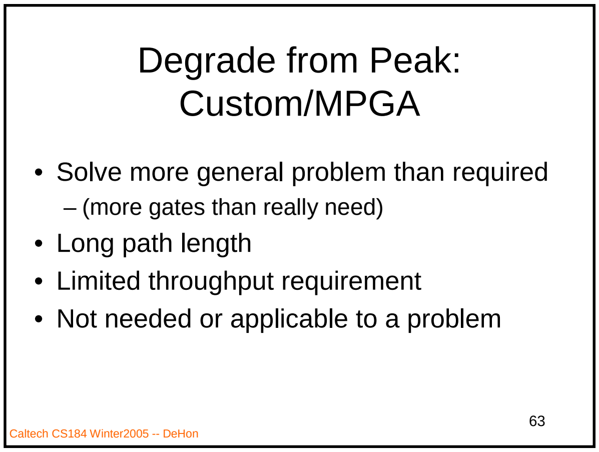
- Long path length → not run at cycle
- Limited throughput requirement
  - – bottlenecks elsewhere limit throughput req.
- Insufficient interconnect
- Insufficient retiming resources (bandwidth)



Degrade from Peak: Processors

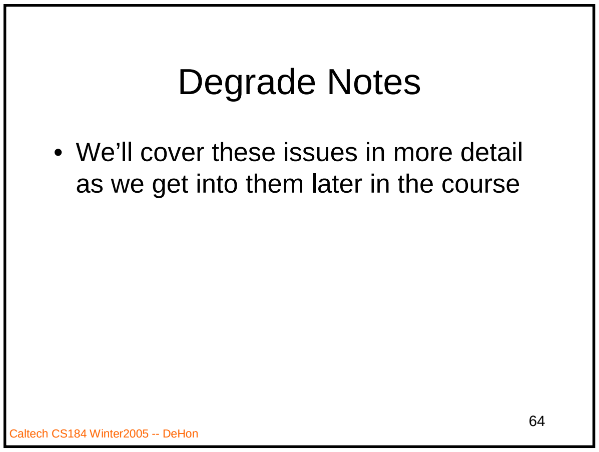
- Ops w/ no gate evaluations (interconnect)
- Ops use limited word width
- Stalls waiting for retimed data

$$\text{E}(\textbf{Functional Density}) = \frac{\textbf{Gate Evaluations}}{\textbf{Datapath Bit}} \times \frac{\textbf{Datapath Bits}}{\textbf{pinst}} \times \frac{\textbf{pinsts}}{\textbf{Issue Slot}} \times \frac{\textbf{Issue Slots}}{\textbf{Clock Cycle}} \times \frac{1}{\textbf{area} \times t_{cycle}}$$



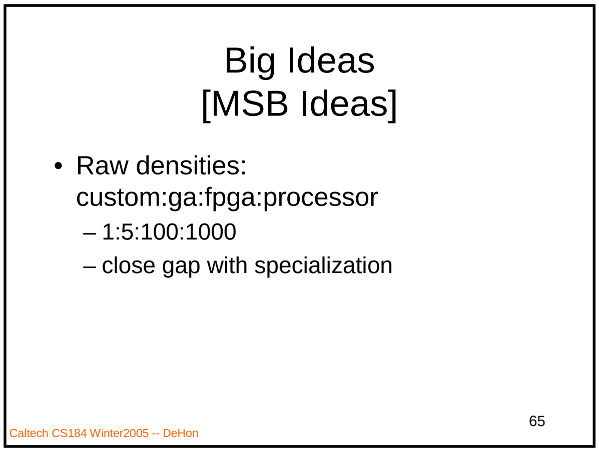
Degrade from Peak: Custom/MPGA

- Solve more general problem than required
  - – (more gates than really need)
- Long path length
- Limited throughput requirement
- Not needed or applicable to a problem



Degrade Notes

- We'll cover these issues in more detail as we get into them later in the course



Big Ideas [MSB Ideas]

- Raw densities: custom:ga:fpga:processor
  - – 1:5:100:1000
  - – close gap with specialization