CS294-7: Reconfigurable Computing

Day 3: Tuesday, January 28

- Admin
- Instructions

Admin

- **Signup for scribe**
- **Class time remains: TR9:30-11** AM
- **Room now: Soda 505**
- *n.b.* **FPGA upcoming (2 weeks)**
  (link to conference program on course calendar)
- **Tuesday, February 11**
  ⇒ **reschedule class to 4:30** PM

Components of a General-Purpose Computation

- **Compute** – generate a result from several inputs (*e.g.* gate, ALU bit, lookup table)
- **Communicate** – move data from producer to consumer
  - **In space** – between producing and consuming processor (*e.g.* bus, crossbar, network)
  - **In time** – producing (arriving) cycle to consuming cycle (*e.g.* register, RF, memory, disk)

Terminology

Primitive Instruction (*pinst*): The collection of bits which tells a bit-processing element how to perform.

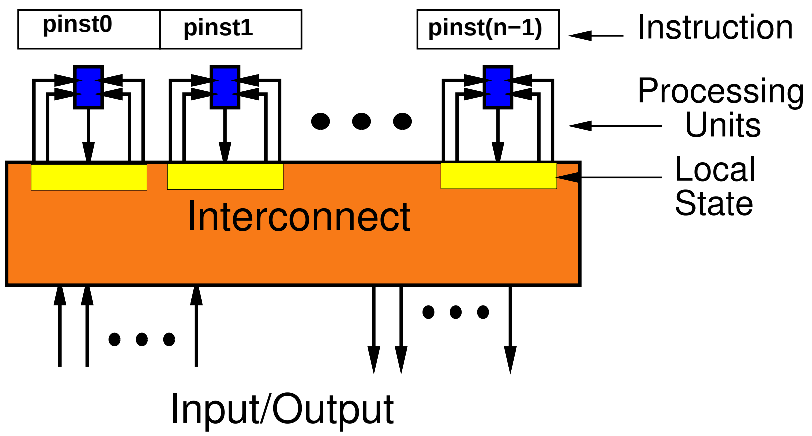
includes:

- **selection of compute operation**
- **input sources in space (interconnect)**
- **input sources in time (retiming)**

Configuration Context: collection of all the bits (all the *pinsts*) which describe the behavior of a general-purpose machine on one operation cycle.

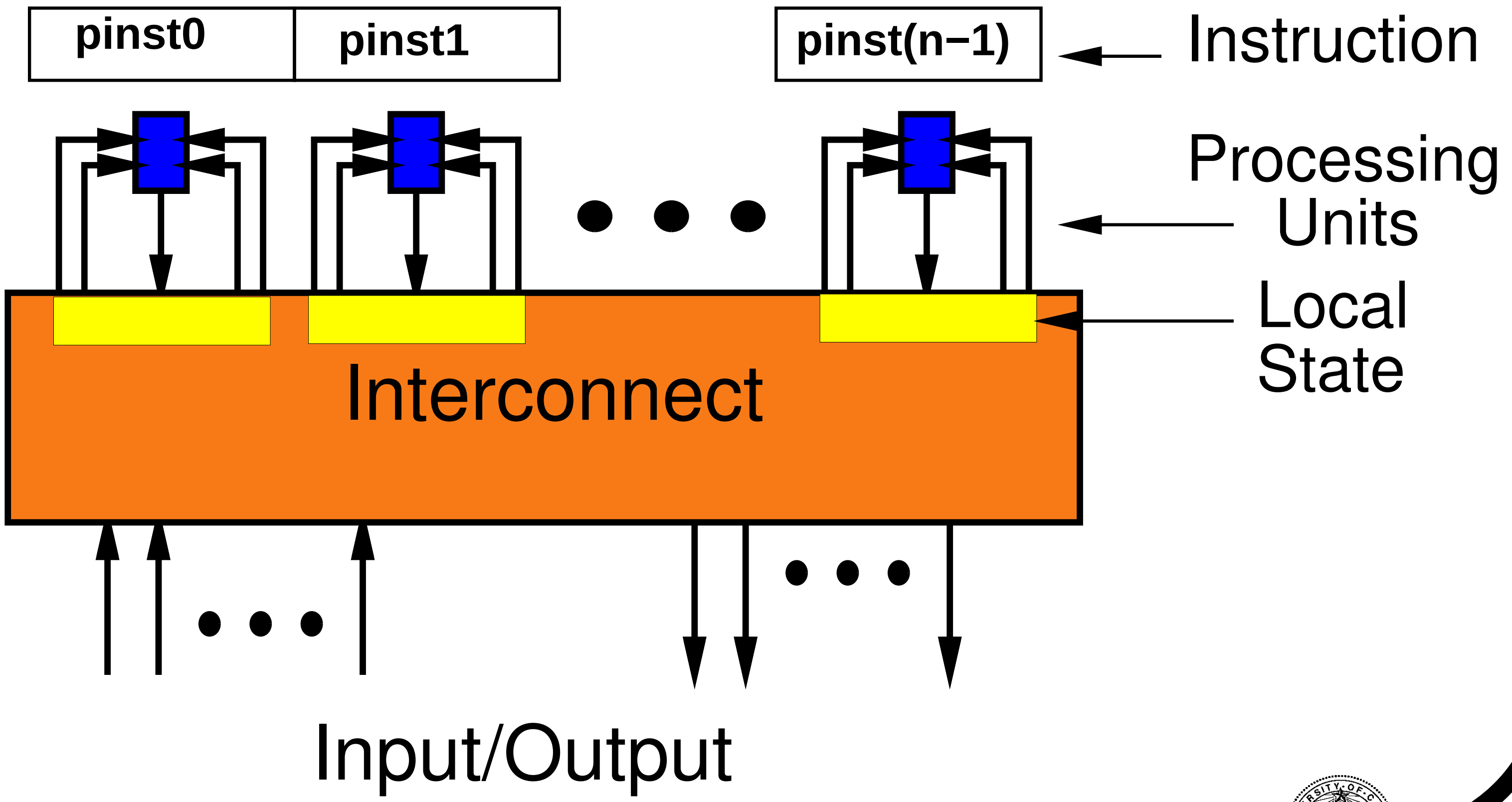
Processing Model

- Collection of Compute Elements
  - Computational Unit
  - Local storage
- Interconnection resources
- *Instruction* controls
  - Compute operation
  - Spatial Interconnect
  - Temporal Interconnect (state)



Ideal Model

Ideal – **Issue a new instruction to every compute element on every cycle**



Ideal BW

Problem: Instruction bandwidth (and memory) can quickly dominate

Consider:

Compute Block	$\approx$ 1Mλ^2 ($\approx$ 1K$\lambda \times$ 1Kλ)
Instruction	$\approx$ 64 bits
Wire pitch	$\approx$ 8λ
Memory bit	$\approx$ 1.2Kλ^2

Ideal BW

2	Room to distribute two instructions across each AE per metal layer (64 bits $\times$ 8λ/bit = 512λ)
$\times$2	Assume two complete metal layers (extreme) distributing in both X and Y
$\times$2	Assume top half of array fed from top side and bottom from bottom side *i.e.* half the array elements from each side

8/AE total across side

Saturate wire channels when: $8 \times \sqrt{N} = N$

64 Processing elements!

Ideal BW

Beyond 64 processors, instruction bandwidth *dictates* **processing element size.**

$$\frac{\sqrt{\textbf{AE (area)}}}{64 \times 8\lambda} \times 4\sqrt{N} = N$$

$$\textbf{AE (area)} = 16K\lambda^2 \times N$$

Ideal Memory Requirements

Idea: Put instruction memory in array

Problem: Instruction memory can quickly dominate

Memory Area = $64 \times 1.2\mathbf{K}\lambda^2$ **/ instruction**

$$A_{AE} = \underbrace{1\mathbf{M}\lambda^2}_{\textbf{base AE (area)}} + \left(N_{inst} \times \underbrace{80\mathbf{K}\lambda^2}_{\textbf{area/instruction}} \right)$$

Pragmatics

Instruction requirements *could* dominate array.

Standard architecture trick:

Look for structure to exploit in "typical computations"

Two Extremes

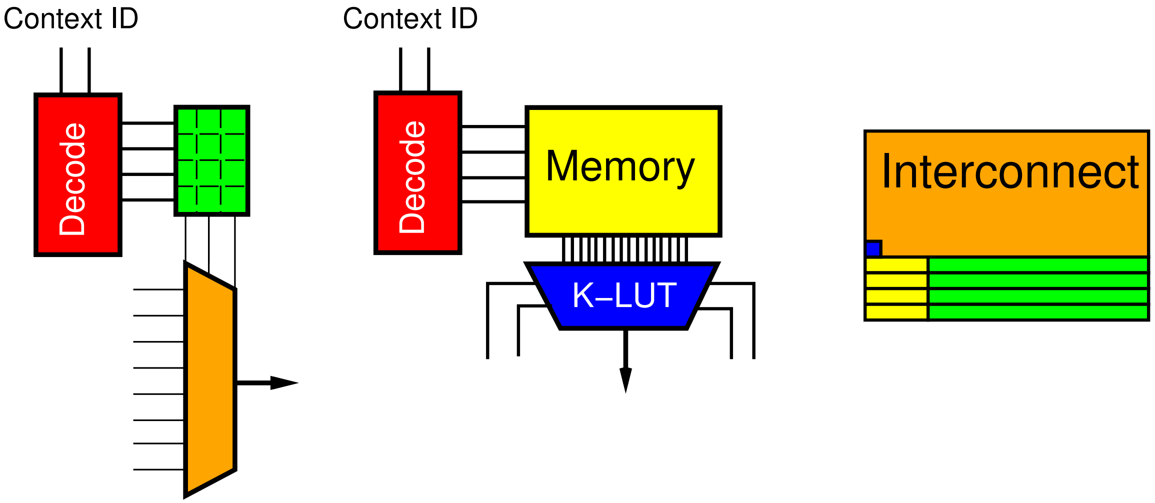
- SIMD Array (most Processors)
  - *Instruction*/cycle
  - share instruction across array
  - uniform in space

- FPGA
  - *Instruction*/AE
  - (extreme) temporal locality
  - uniform in time
  - Slow “programming” phase

Hybrids

- VLIW (Superscalar)
  - Few *pinsts*/cycle
  - Share instruction across w bits

- DPGA
  - Small instruction-store/AE



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, Programmable Cellular Automata
			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	1	c	$n_v \cdot w$	MSIMD
m	1	c	1	VEGA
		8	16	PADDI-2
		c	w	MIMD (traditional)

Architectural Parameter Implications

What happens when:

- Datapath is too wide?
- Datapath is too narrow?
- Instruction memory is too deep?
- Instruction memory is too shallow?

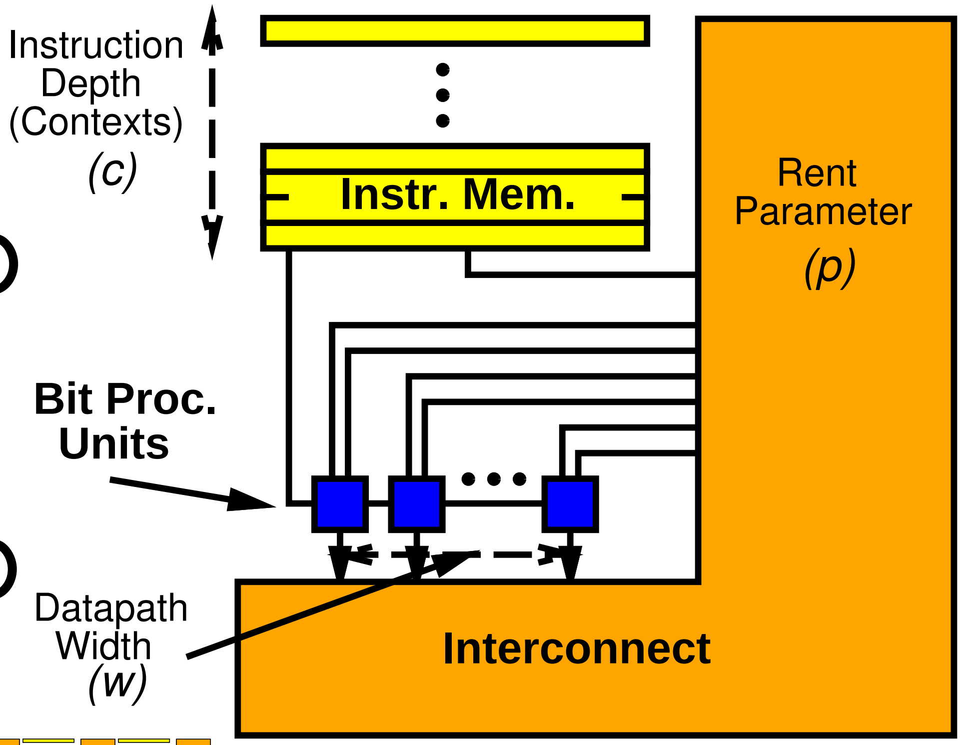
A Model

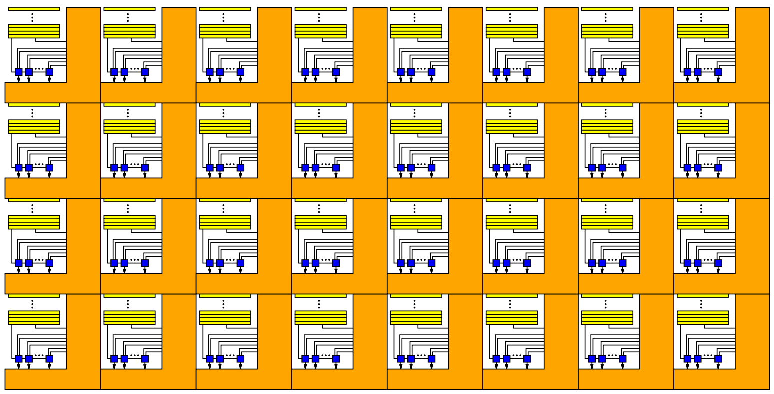
- Device Area
- Device Utilization

Computing Device Composition

Composition:

- **Bit Processing Units**
  (*e.g.* ALU bit, 4-LUT)
- **Interconnect**
  - Spatial
  - Temporal (retiming)
- **Instruction Memory**





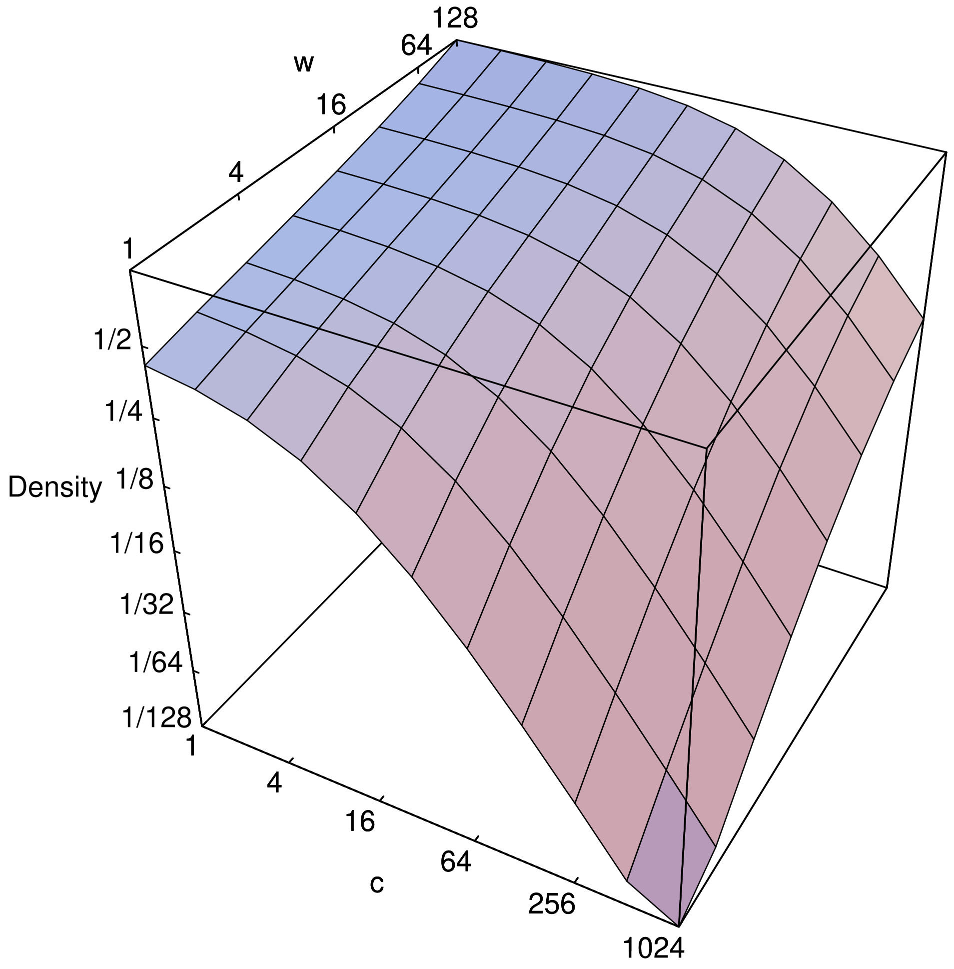
Model Area

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

Spot Check Model

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

Peak Computational Density from Model



(at 16K Bit Processing Elements)

Empirical Peak Computational Densities

Year	Design	Organization	λ^2 area	cycle	$\frac{ge's}{\lambda^2 \cdot s}$
Microprocessors					
1987	MIPS-X	1×32	68M	50ns	19
1995	Alpha	2×64	4.8G	3.3ns	18
Vector					
1995	T0	$1\times32+16\times32$	1.1G	22ns	45
Reconfigurable ALUs					
1995	PADDI-2	48×16	515M	20ns	150
FPGAs					
1992	Xilinx 4K	49 CLBs (2 4-LUT)	61M	7ns	230
SIMD					
1995	Abacus	1000 (2 3-LUT)	190M	8ns	660

Efficiency

Architectural parameters (c, w) have duals in tasks.
Peak efficiency when matched.

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

E.g.

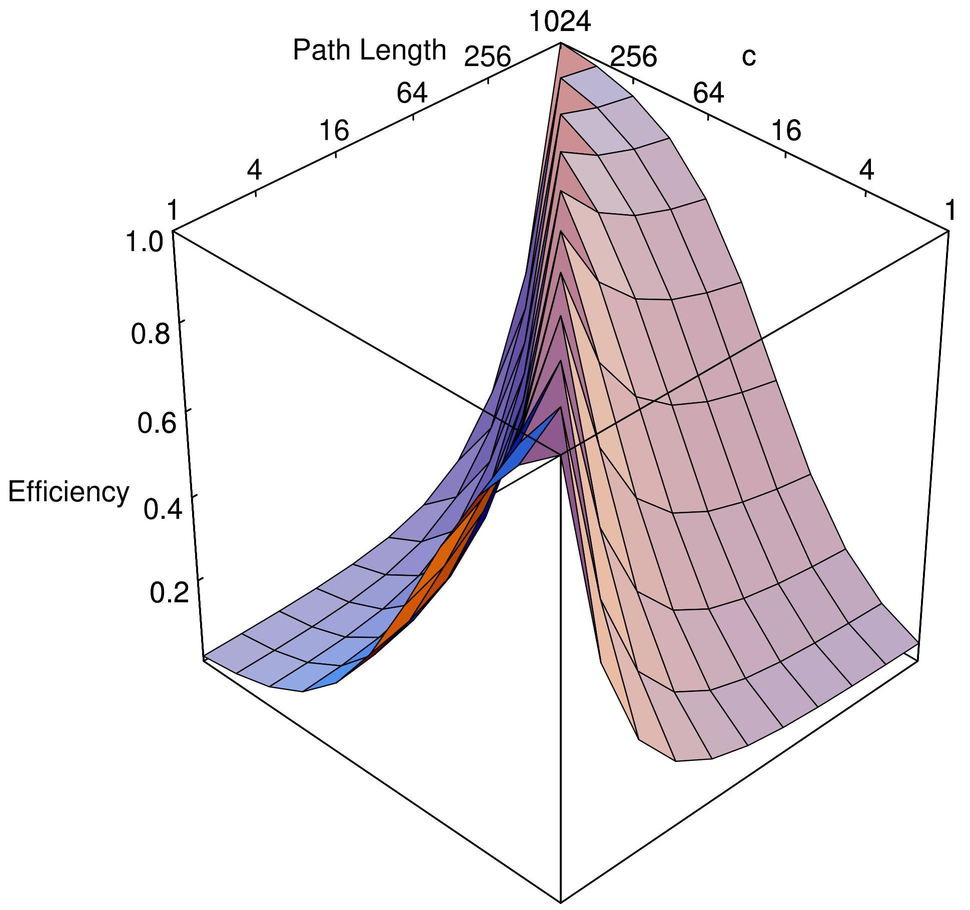
If $w_{task} > w_{arch}$**:**

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

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

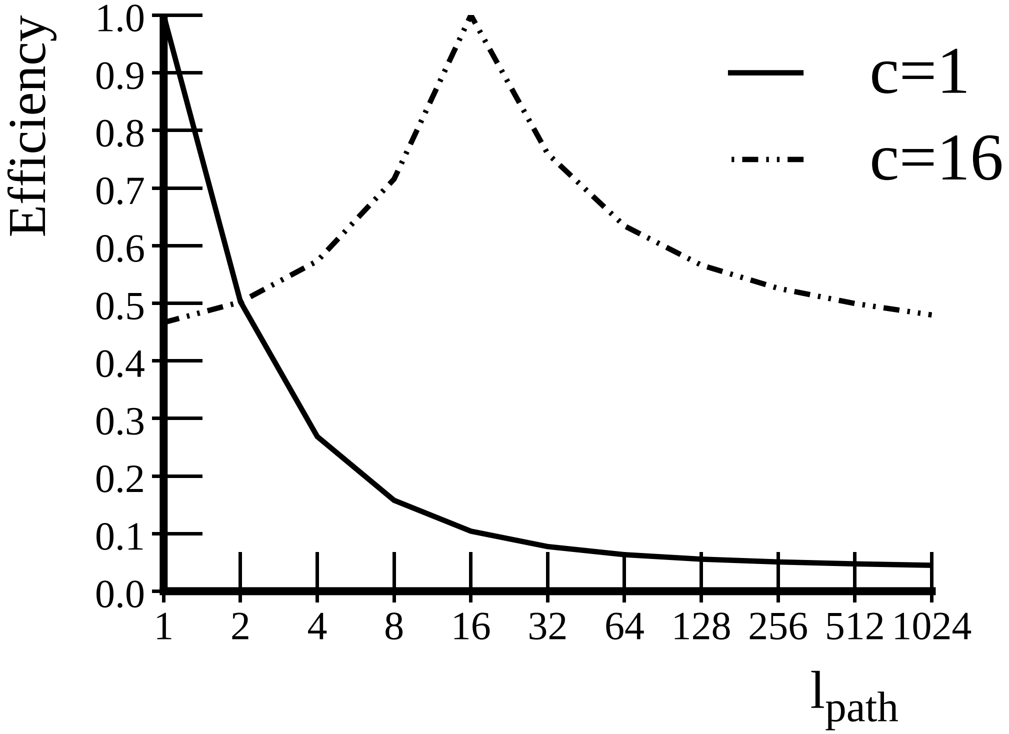
$$\textbf{Efficiency} = \frac{w_{task} \times A_{bit\_elm|w=w_{task}}}{w_{arch} \times A_{bit\_elm|w=w_{arch}}}$$

Efficiency with fixed Width ($w = 1$)



(at 16K Bit Processing Elements)

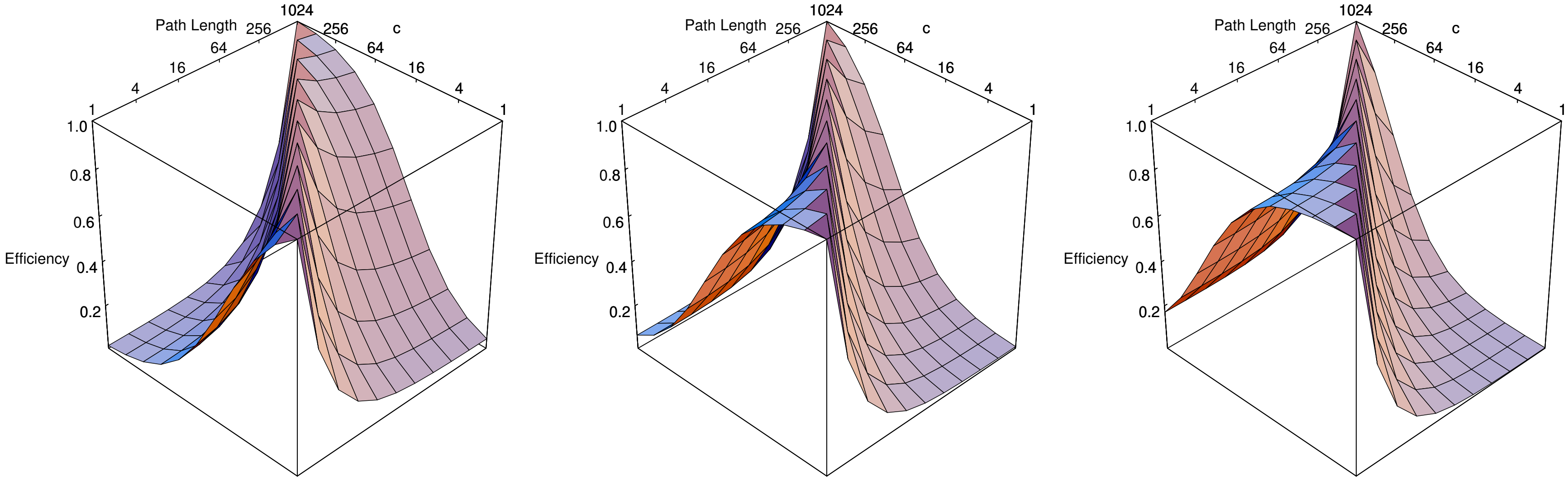
Pedagogical FPGA v/s DPGA Comparison



Robust point:

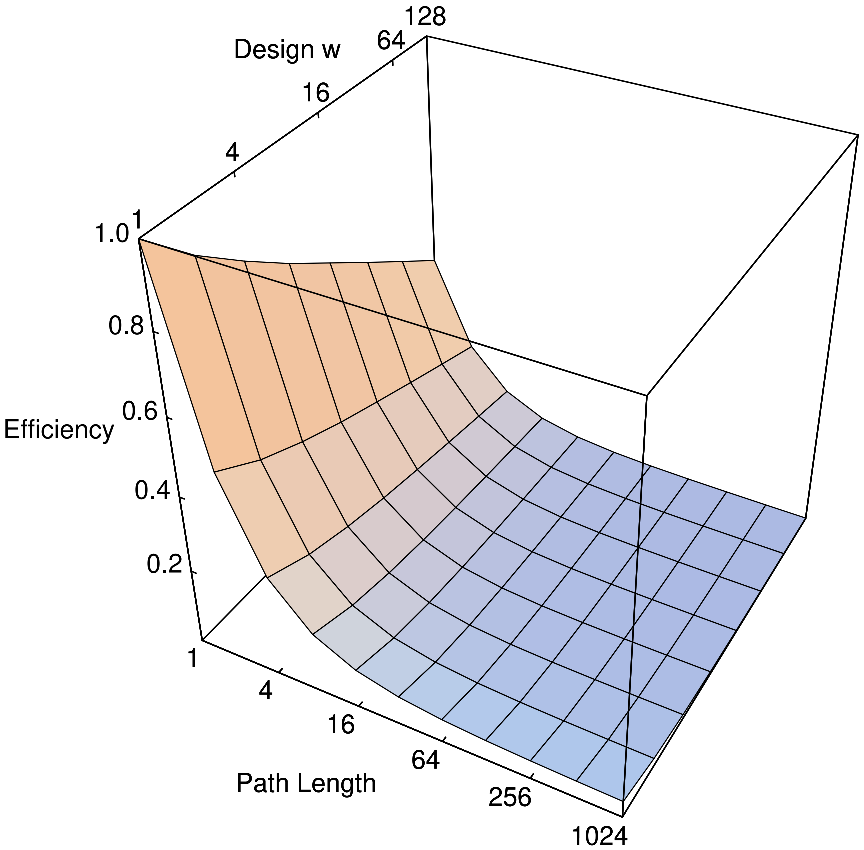
$$A_{bit\_elm}|_{c=0} \approx \left(\frac{1}{2}\right) A_{bit\_elm}|_{c=c^*}$$

Robust Point Changes with Datapath Width



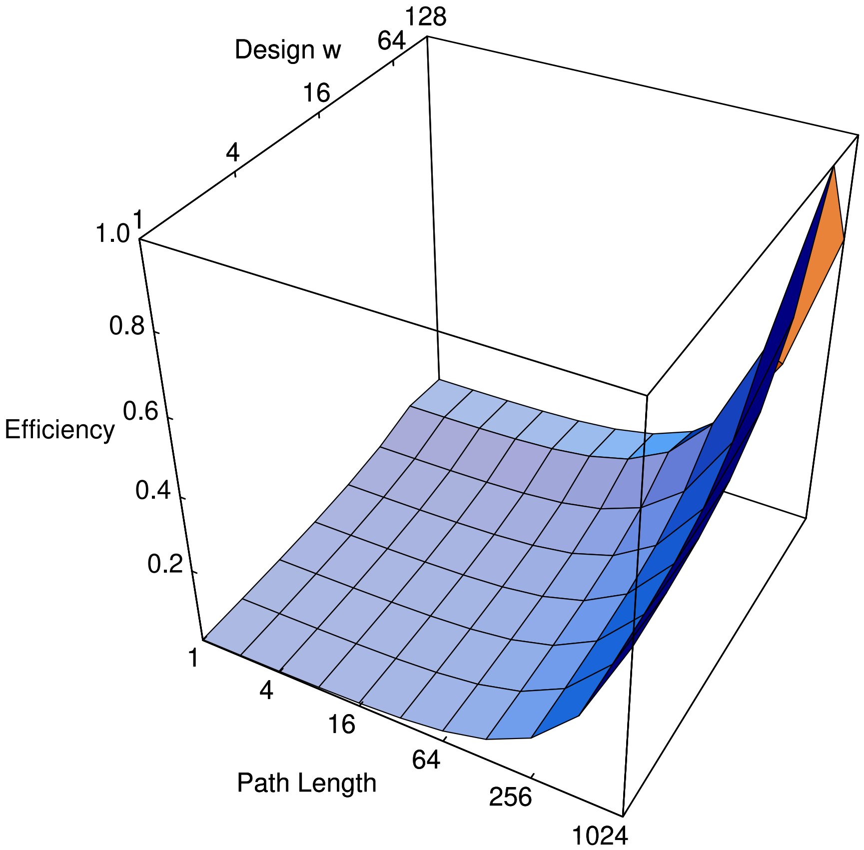
Left – $w = 1$**; Middle –** $w = 8$**; Right –** $w = 64$

Comparing “Processors” and FPGAs



FPGA

$c = d = 1,\ w = 1$

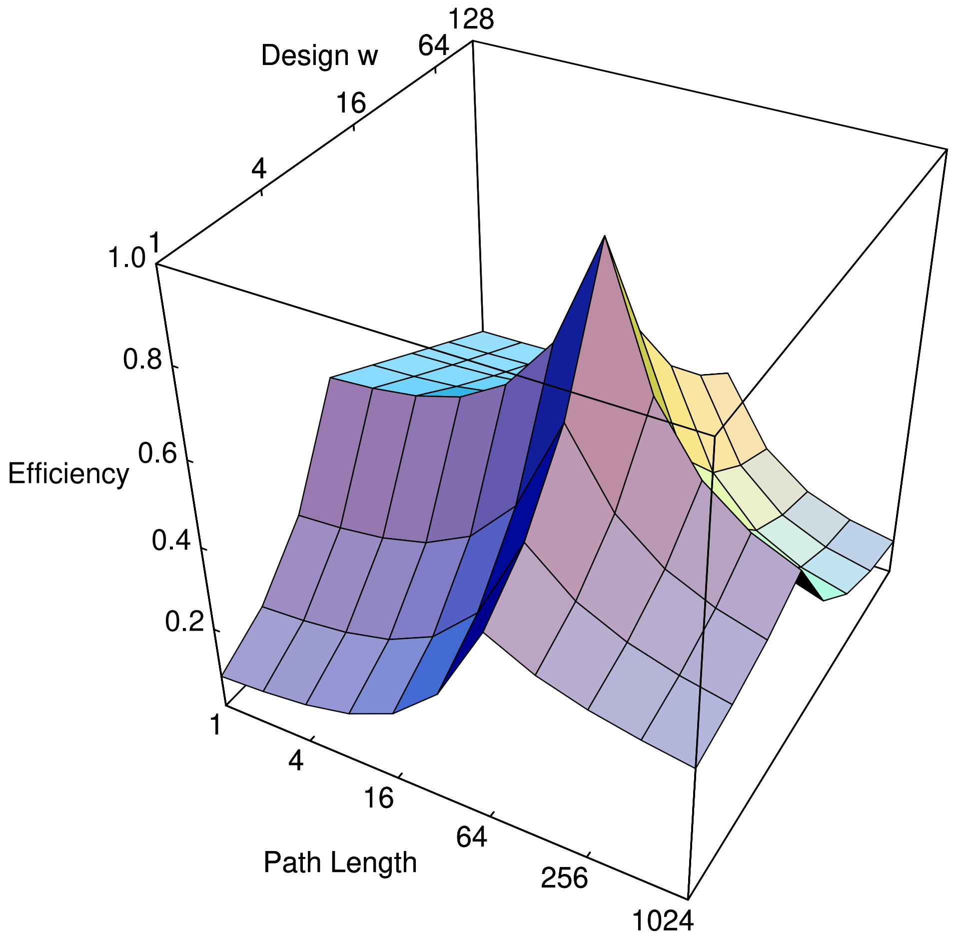


“Processor”

$c = d = 1024,\ w = 64$

(at 16K Bit Processing Elements)

Efficiency of Intermediate Architecture



(at 16K Bit Processing Elements)

$$w = 8, \; c = 64$$

Caveats

Abstracting away many details

Notably:

- interconnect
- control
- specialized functional units

Applications are heterogeneous mix of characteristics.

Architectural space is large.

Review

- Instruction resources can be significant
- Exploit application structure
  to reduce instruction requirements
- Classify architectures by instruction organization
- Architecture/application mismatches $\rightarrow$ inefficiencies
- Area model $\rightarrow$ visualize efficiency trends