MATRIX
A Reconfigurable Computing Device with Configurable Instruction Distribution

André DeHon <andre@acm.org>
Dan Hartman <omv@ai.mit.edu>
Ethan Mirsky <eamirsky@ai.mit.edu>

Reinventing Computing
MIT Artificial Intelligence Laboratory

Acknowledgments: This research is supported by the Advanced Research Projects Agency of the Department of Defense under Rome Labs contract number F30602-94-C-0252.

MIT AI Lab

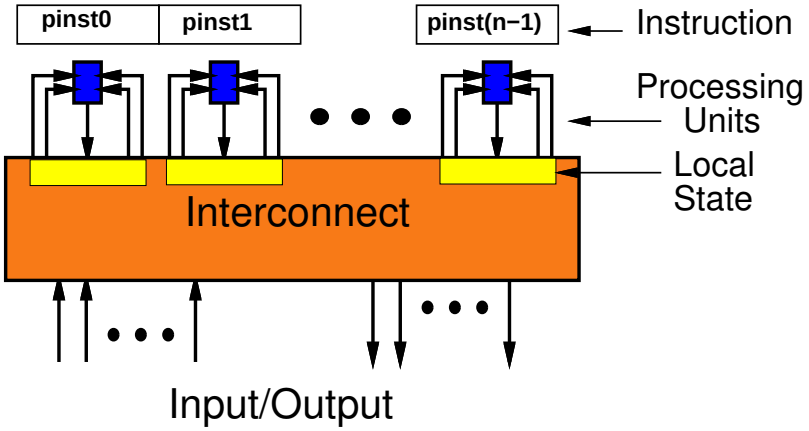
Outline

- **Instructions**
  - Processing Model
  - Instruction Bandwidth and Memory
  - Typical Restrictions
  - Architecture Taxonomy
  - Implications
- **MATRIX**
  - Problem Identification
  - Benefits of Configurable Instruction Distribution
  - MATRIX Architecture
  - Usage Example – Convolution
  - Implementation Details
  - Summary

MIT AI Lab

Processing Model

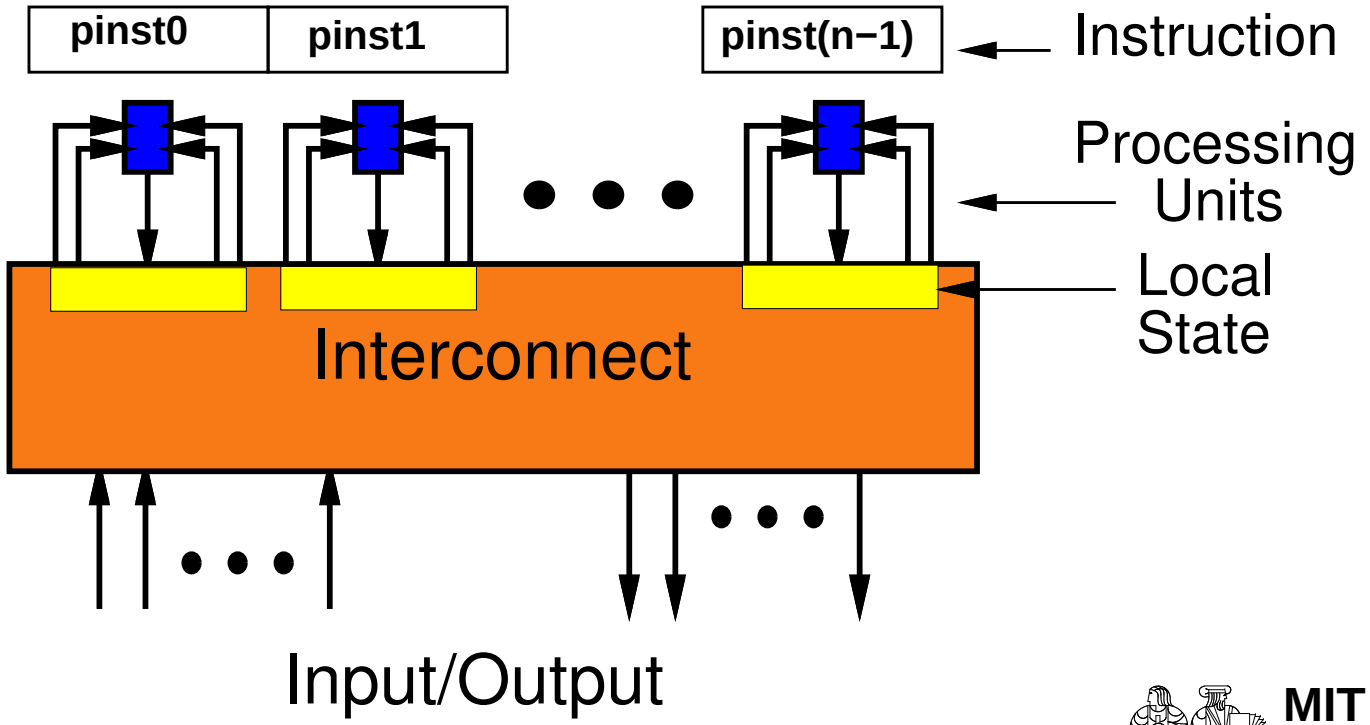
- Collection of Compute Elements
  - Computational Unit
  - Local storage
- Interconnection resources
- *Instruction* controls
  - Compute operation
  - Spatial Interconnect
  - Temporal Interconnect (state)



pinst – collection of bits which tells a bit-processing element how to perform.

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:

$$\begin{aligned} \text{Compute Block} &\approx 1\text{M}\lambda^2 \ (\approx 1\text{K}\lambda \times 1\text{K}\lambda) \\ \text{Instruction} &\approx 64 \text{ bits} \\ \text{Wire pitch} &\approx 8\lambda \\ \text{Memory bit} &\approx 1.2\text{K}\lambda^2 \end{aligned}$$

MIT AI Lab

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!

MIT AI Lab

Ideal BW

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

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

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

MIT AI Lab

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**

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

MIT AI Lab

Pragmatics

Instruction requirements *could* dominate array.

Standard architecture trick:

Look for structure to exploit in "typical computations"

MIT AI Lab

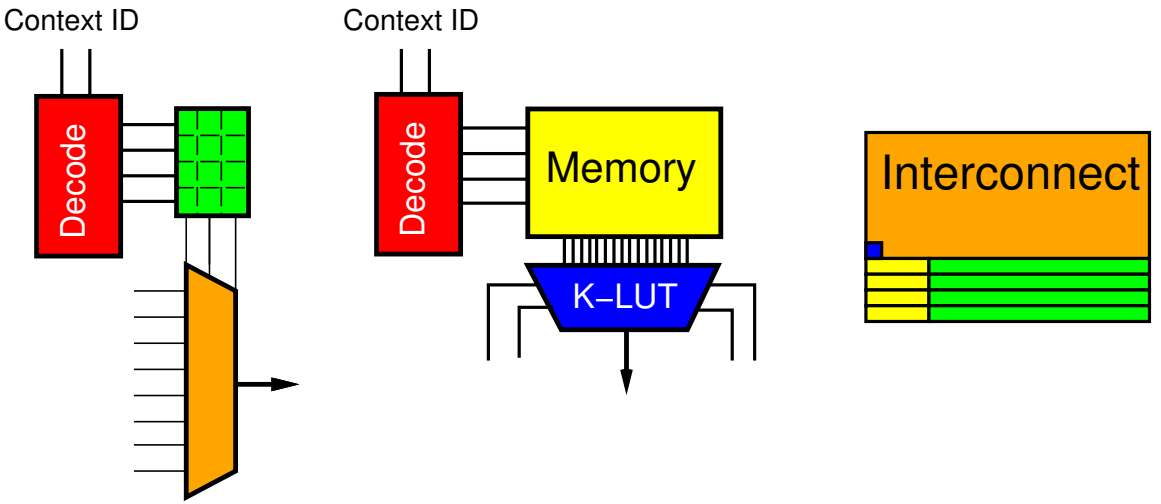
Two Extremes

- **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**

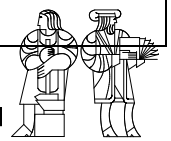
Hybrids

- 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)
			1	FPGA, Programmable Cellular Automata
	n	1	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)



MIT AI Lab

Architectural Parameter Implications

What happens when:

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

MIT AI Lab

A Model

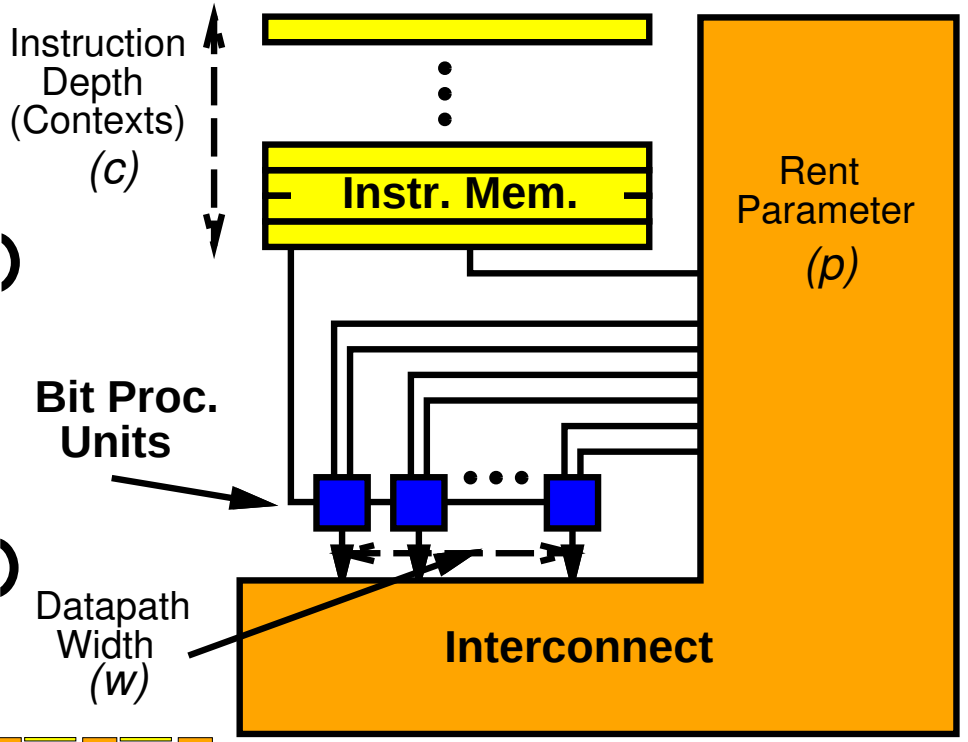
- Device Area
- Device Utilization

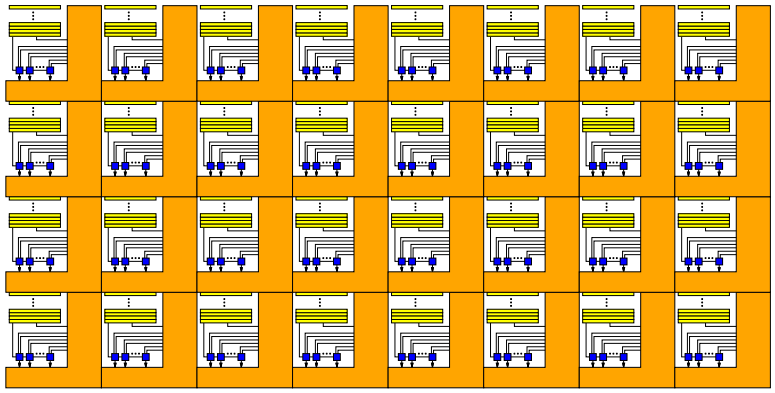
MIT AI Lab

Computing Device Composition

Composition:

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





Model Area

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

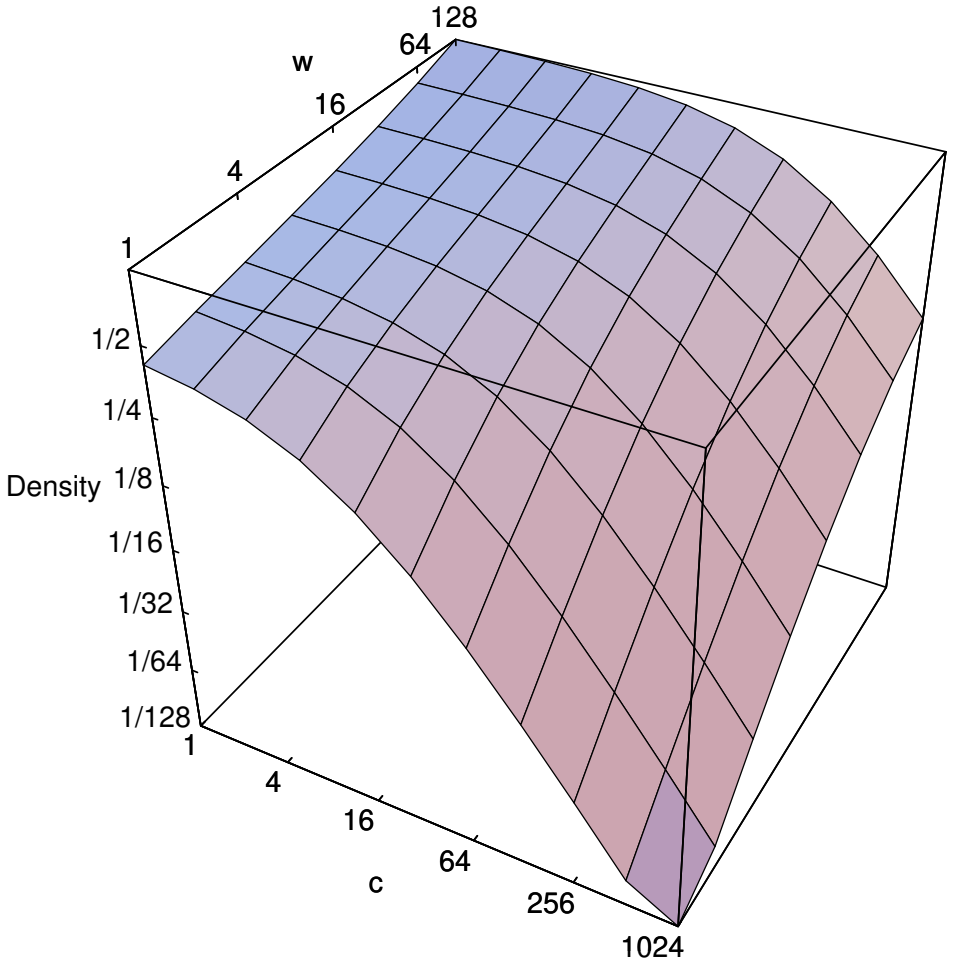
MIT AI Lab

Spot Check 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

MIT AI Lab

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	$1 \times 32 + 16 \times 32$	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

MIT AI Lab

Efficiency

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

$$\textbf{Efficiency} = \frac{A_{matched_arch}}{A_{arch}}$$

E.g.

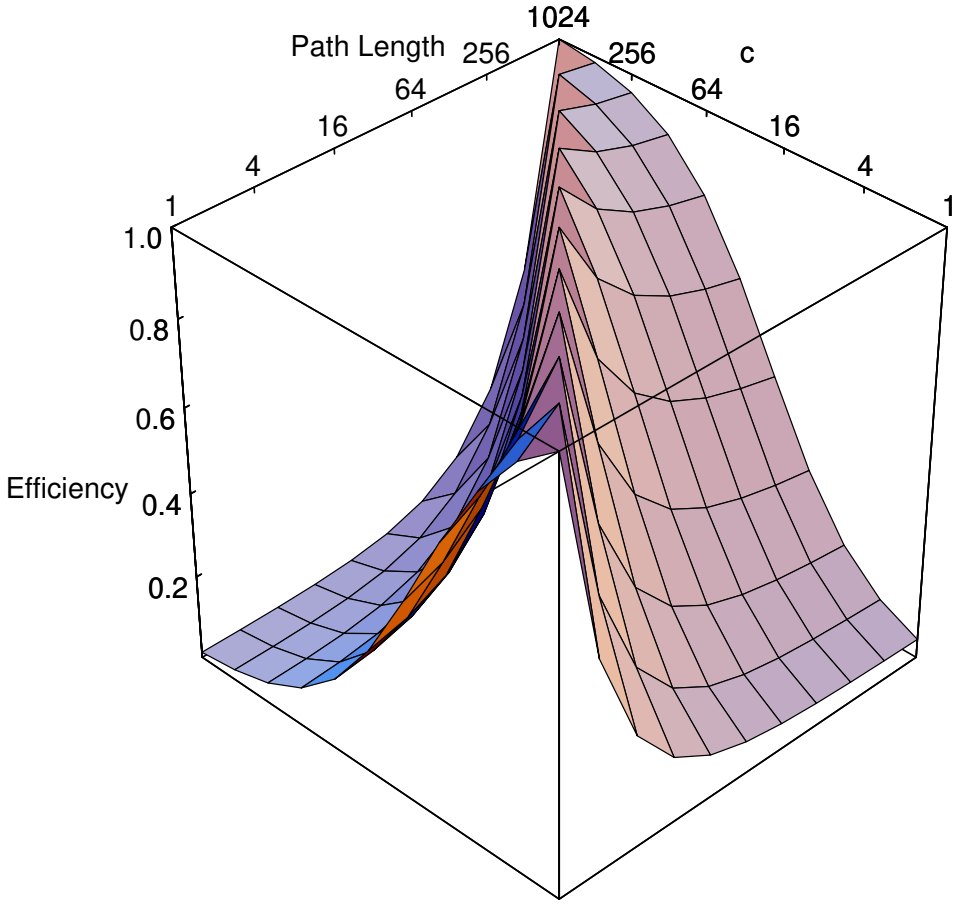
If $w_{task} > w_{arch}$**:**

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

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

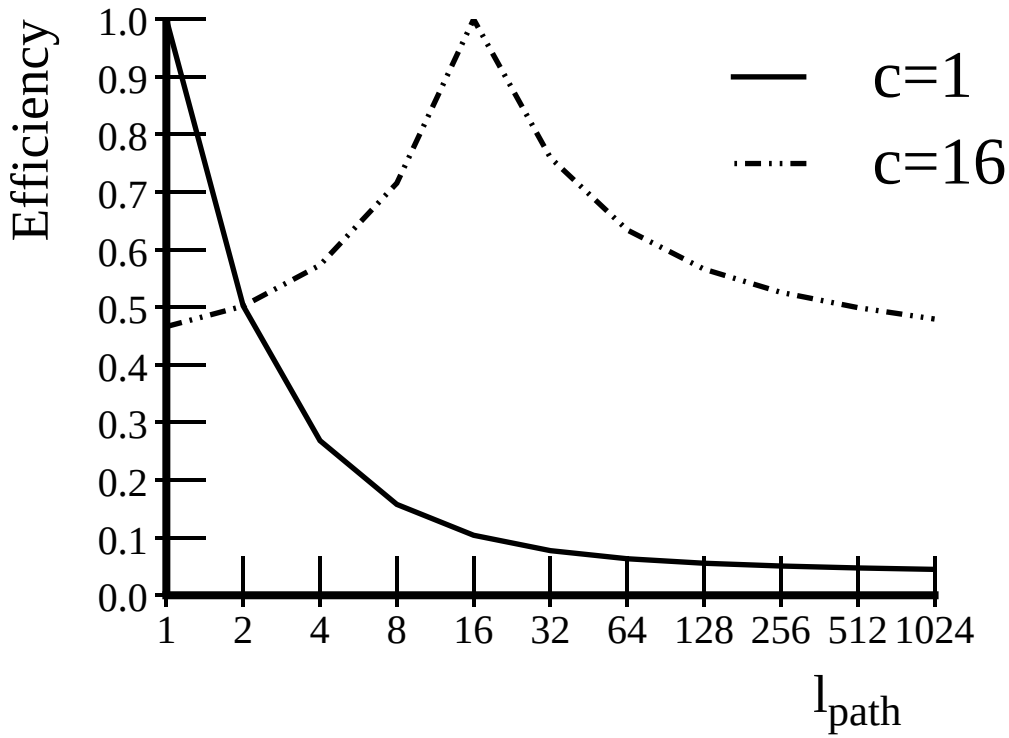
$$\textbf{Efficiency} = \frac{w_{task} \times A_{bit_elm}|w=w_{task}}{w_{arch} \times A_{bit_elm}|w=w_{arch}}$$

Efficiency with fixed Width ($w = 1$)



(at 16K Bit Processing Elements)

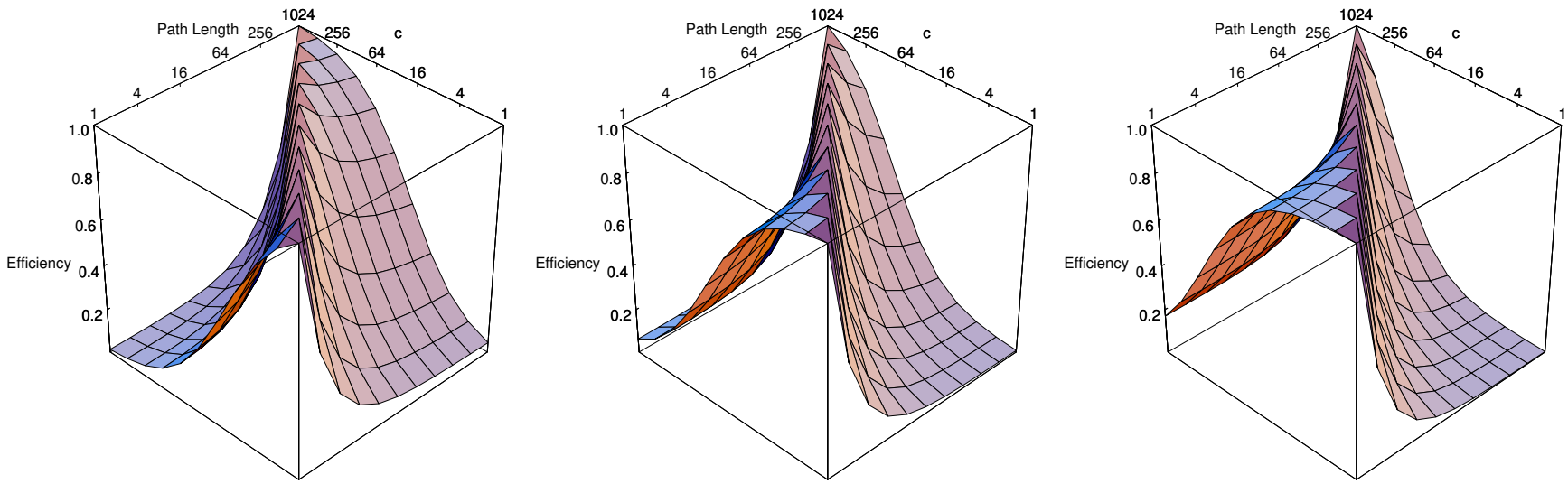
Pedagogical FPGA v/s DPGA Comparison



Robust point:

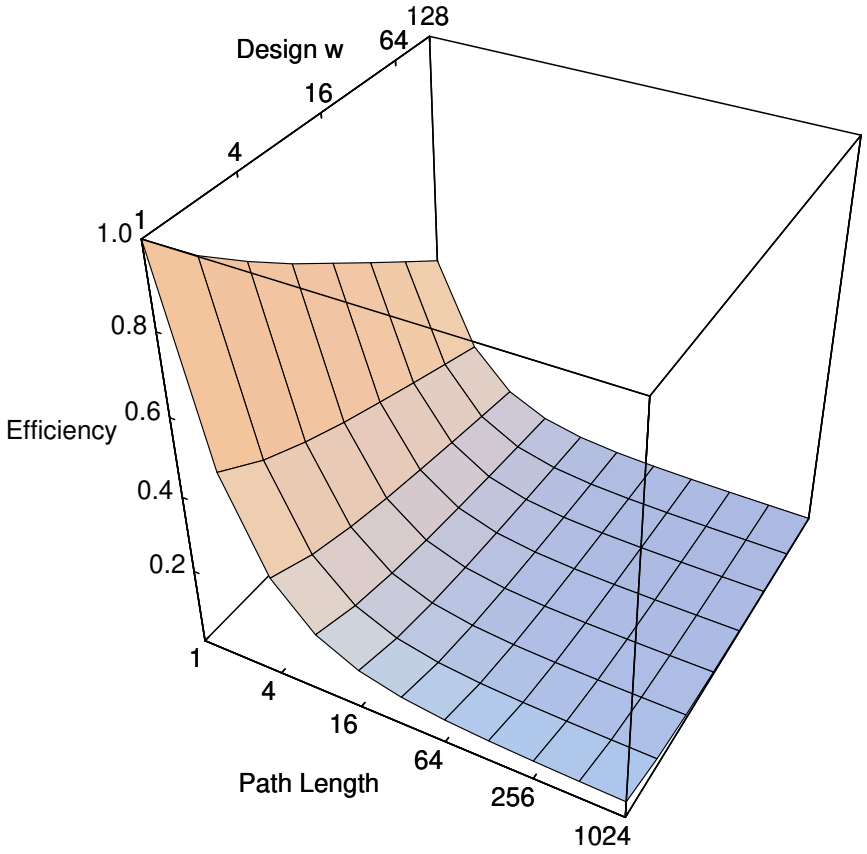
$$A_{bit_elm}|_{c=0} \approx \left(\frac{1}{2}\right) A_{bit_elm}|_{c=c^*}$$

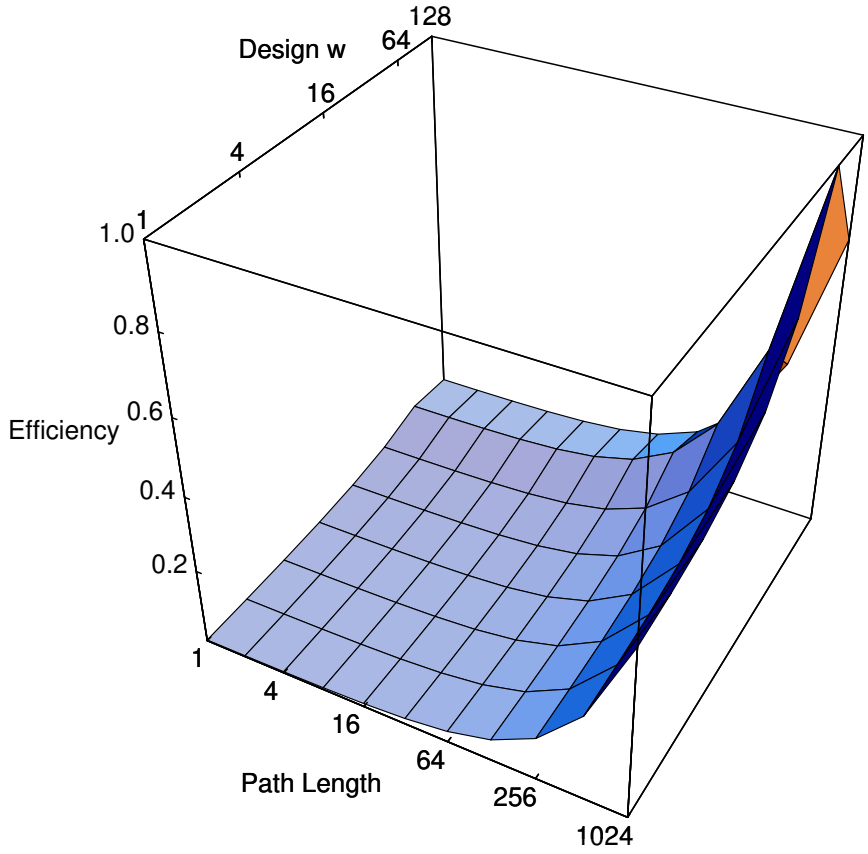
Robust Point Changes with Datapath Width



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

Comparing "Processors" and FPGAs





FPGA

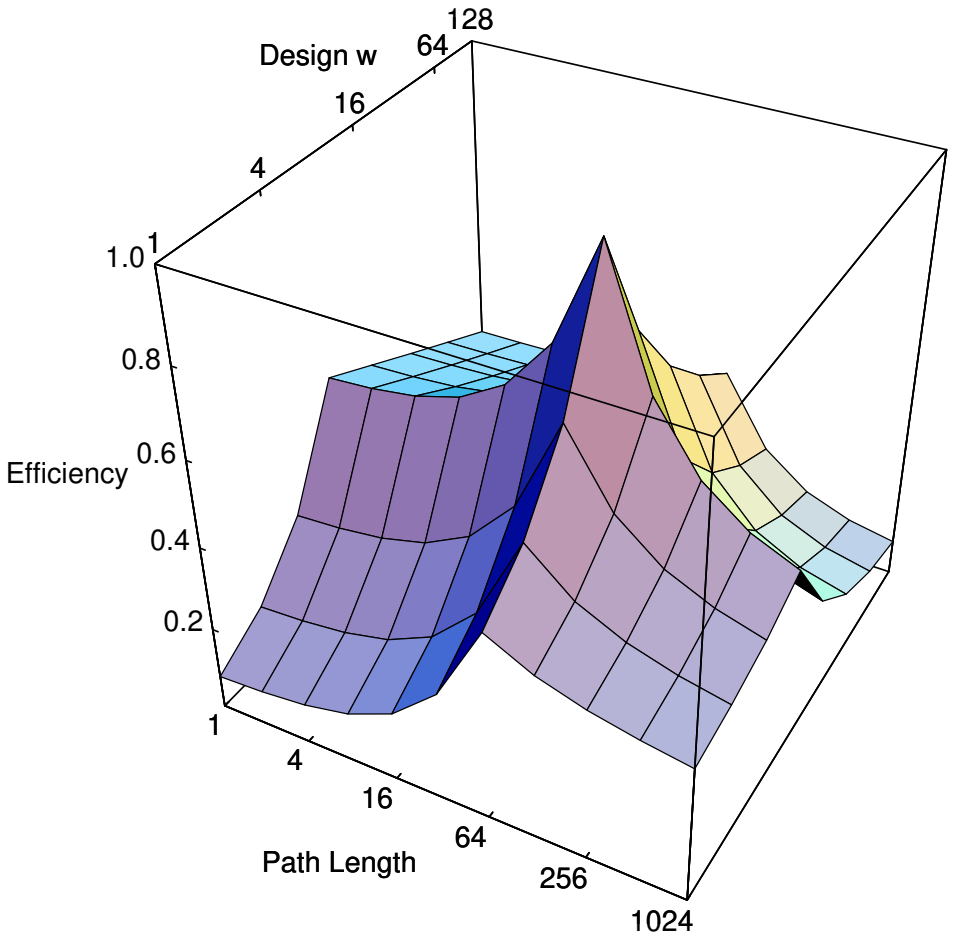
$c = d = 1$, $w = 1$

"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.

MIT AI Lab

Instruction 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

Outline

- **Instructions**
  - Processing Model
  - Instruction Bandwidth and Memory
  - Typical Restrictions
  - Architecture Taxonomy
  - Implications
- **MATRIX**
  - Problem Identification
  - Benefits of Configurable Instruction Distribution
  - MATRIX Architecture
  - Usage Example – Convolution
  - Implementation Details
  - Summary

MATRIX

So, where does MATRIX come in?

$\Rightarrow$ By:

1. picking a robust point for primitive resources
2. making instruction distribution configurable

MATRIX is efficient across a wider range of the application space.

The Problem

We want to have:

- Computing solutions which are programmable
  (post fabrication and in-system)

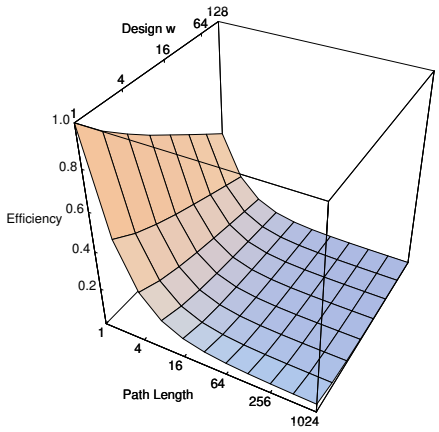
- Efficient (application oriented) Silicon Usage
  (minimize size and cost necessary to solve problem)

MIT
AI
Lab

Traditional Approaches

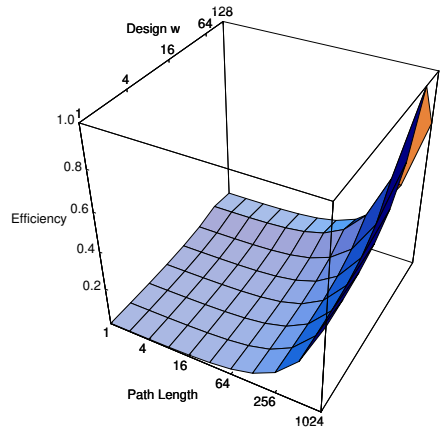
Traditional Solutions (*e.g.* Processors and FPGAs)

- Fix Instruction Distribution
- Limits application range where architecture is efficient



FPGA

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



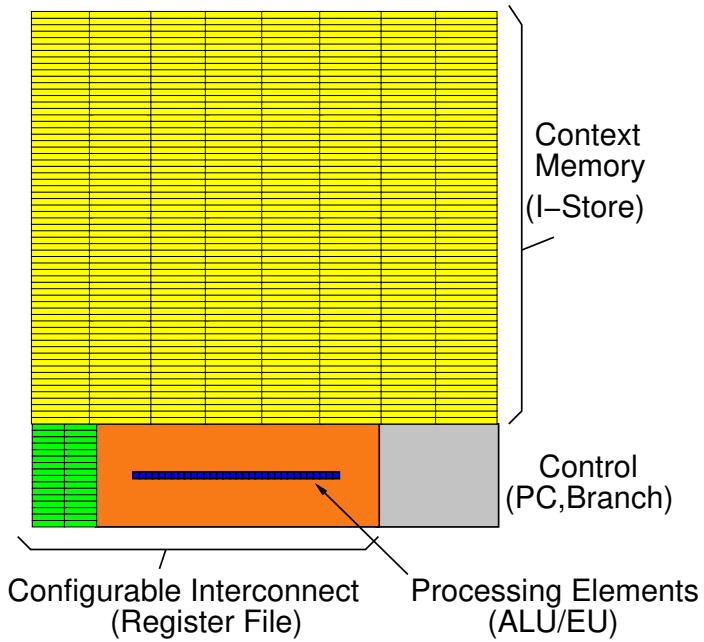
"Processor"

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

Processors

Dedicate significant area to instruction and data memory

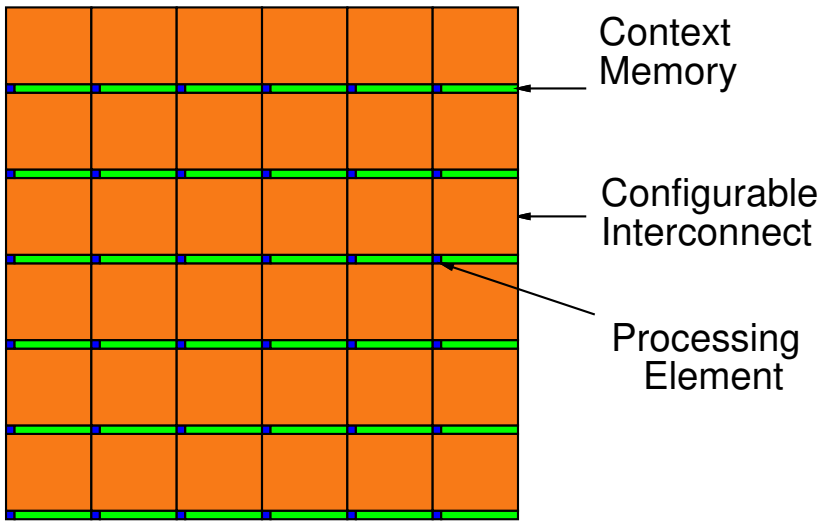
- ⇒ efficient for heavily multiplexed operation
- ⇒ inefficient on regular tasks
- ⇒ lower yielded capacity on fine-grained data



Field-Programmable Gate Arrays

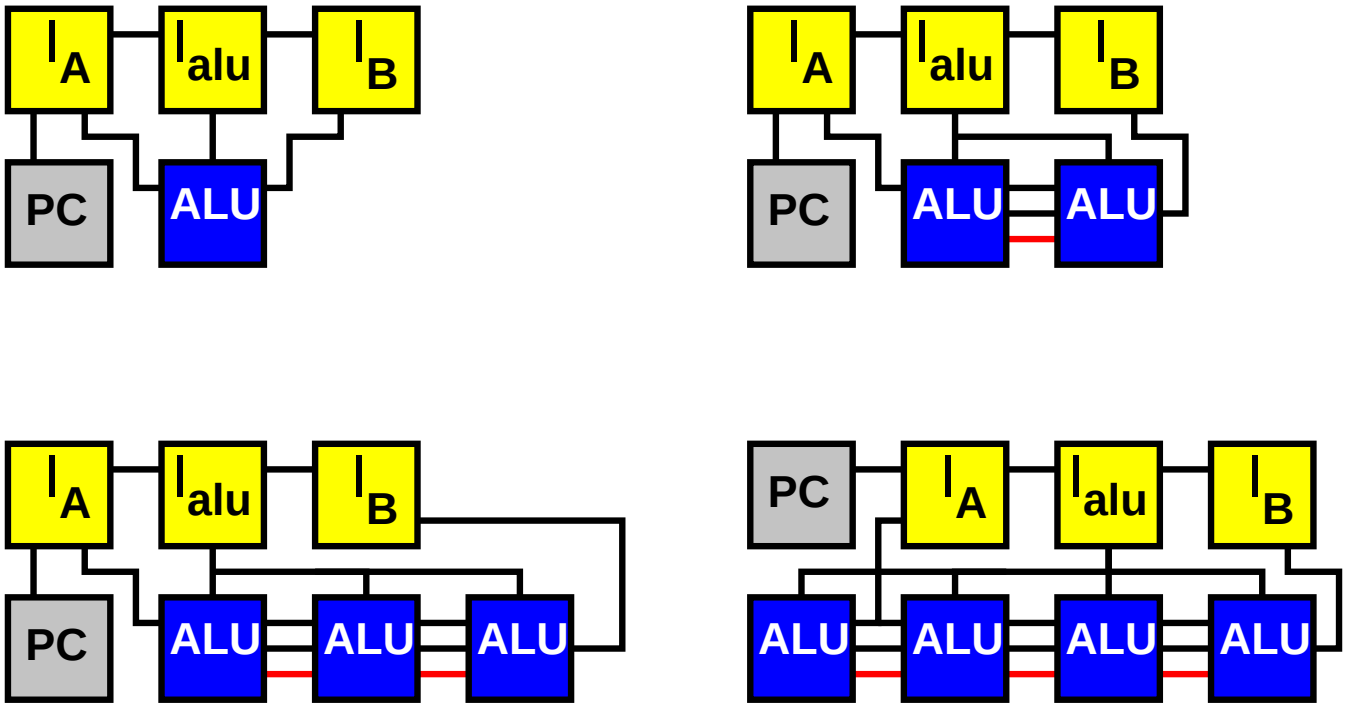
Dedicate most area to fine-grained interconnect

- ⇒ efficient on very regular operations
- ⇒ high fine-grain yield
- ⇒ low yielded capacity on irregular tasks

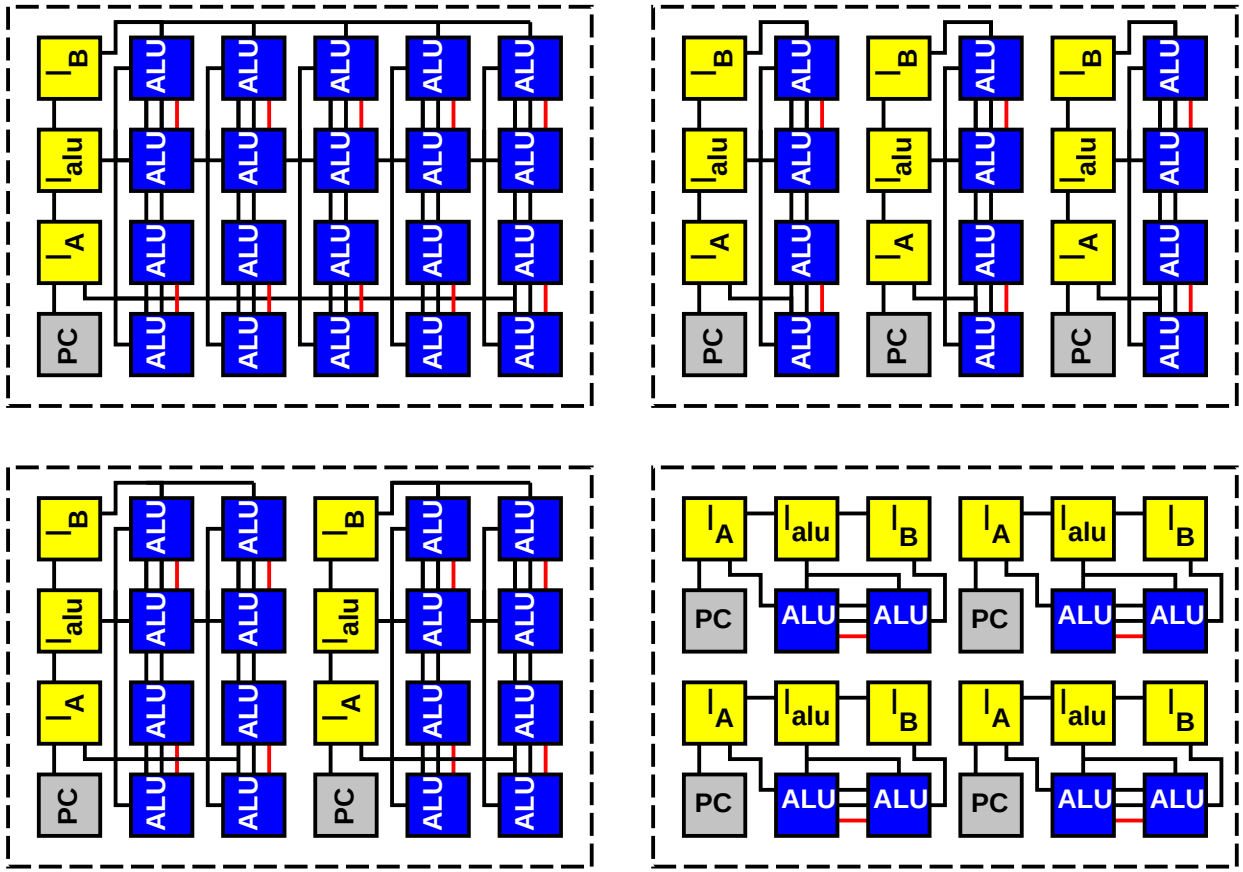


MATRIX Benefits: Configurable Granularity

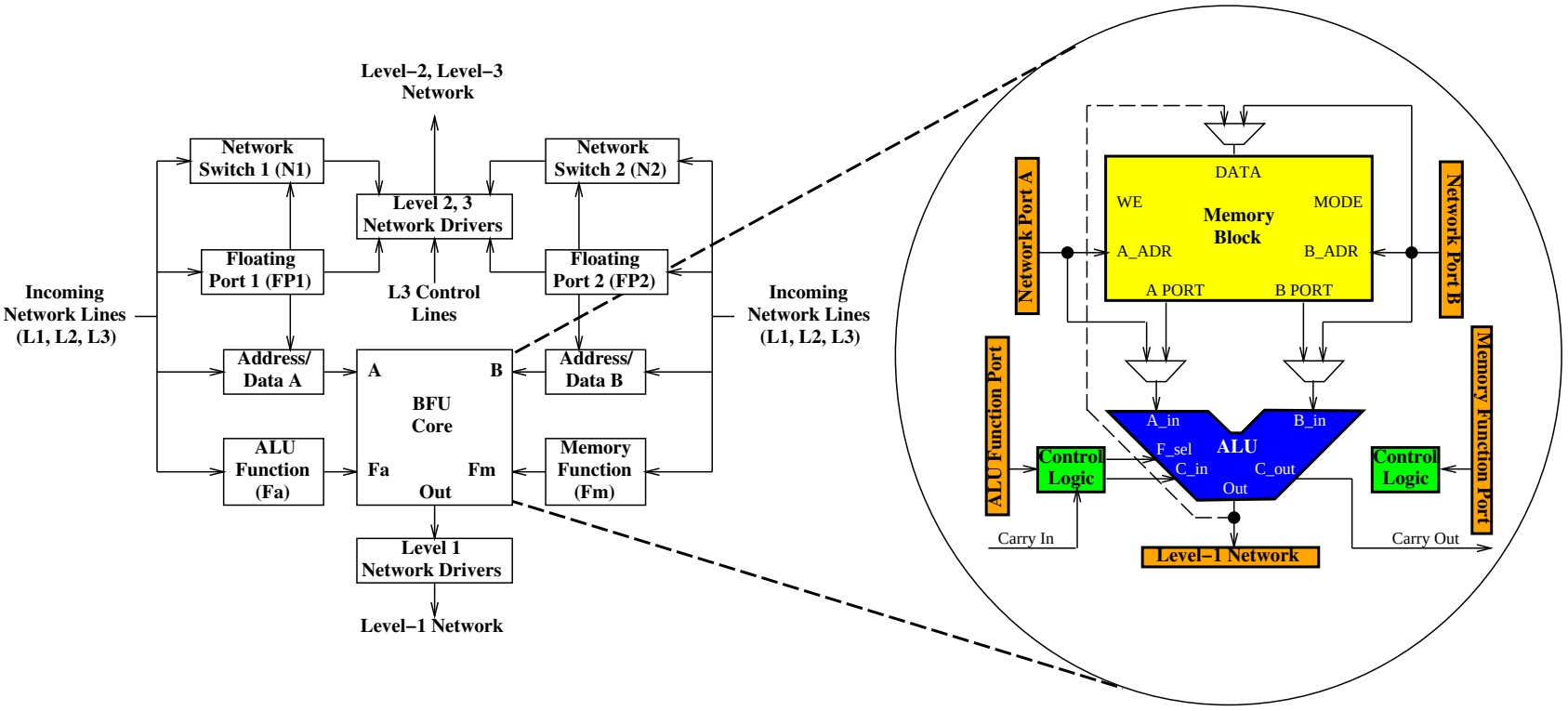
Select granularity according to needs of the application.



MATRIX Benefits: Configurable Instruction Distribution



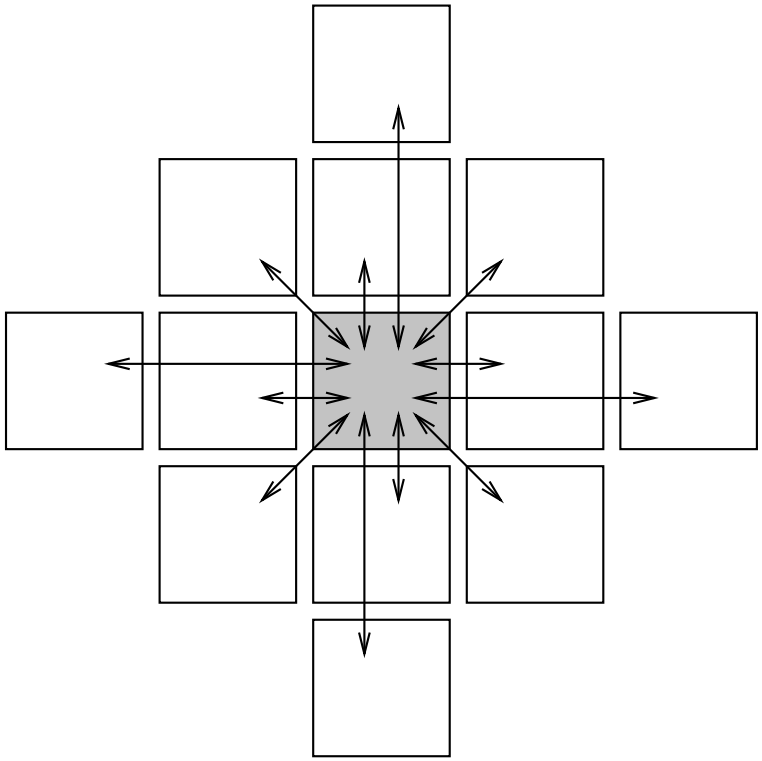
MATRIX Architecture
Basic Functional Unit (BFU)



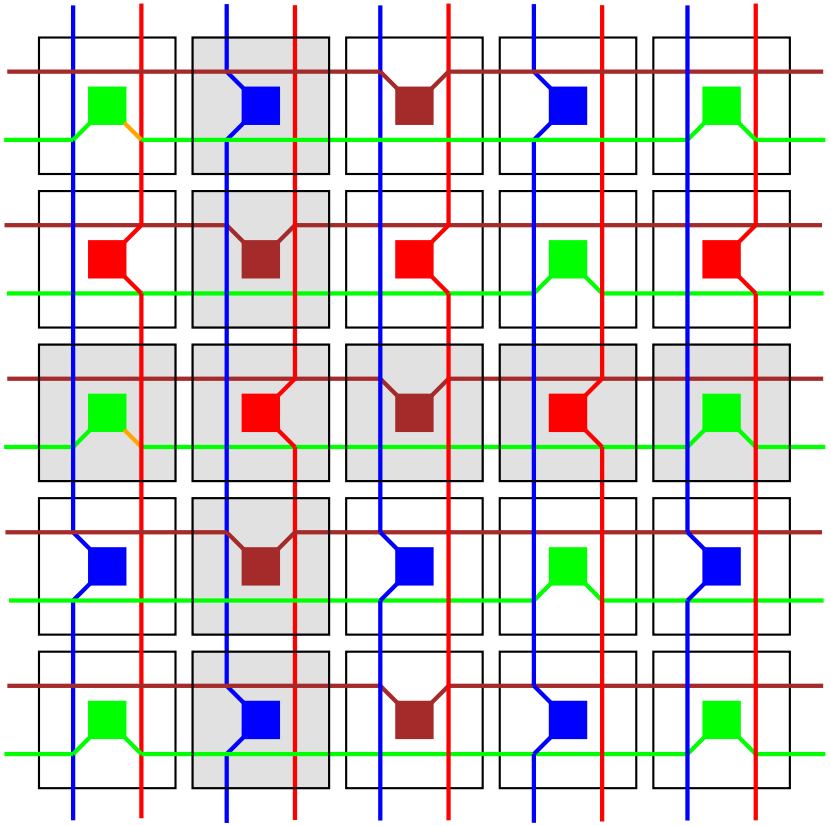
- Same network servers data, instructions, and addresses
- Can use BFU as Instruction Store, Data Memory, or Compute Unit

MATRIX Architecture (Network)

Nearest Neighbor Interconnect

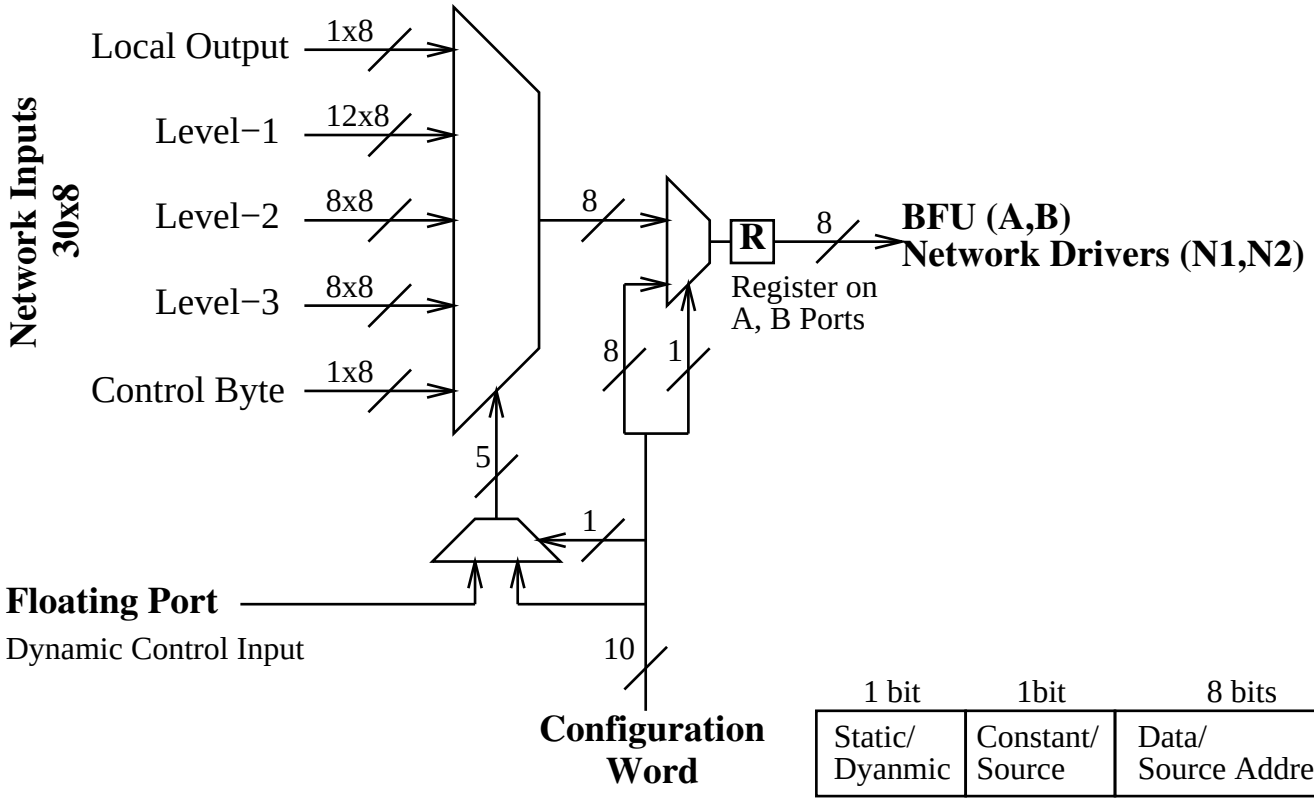


Length Four Bypass Interconnect



Global – 4 Lines per Row/Column

MATRIX Port Architecture



Usage Example
Finite Impulse Response Computation

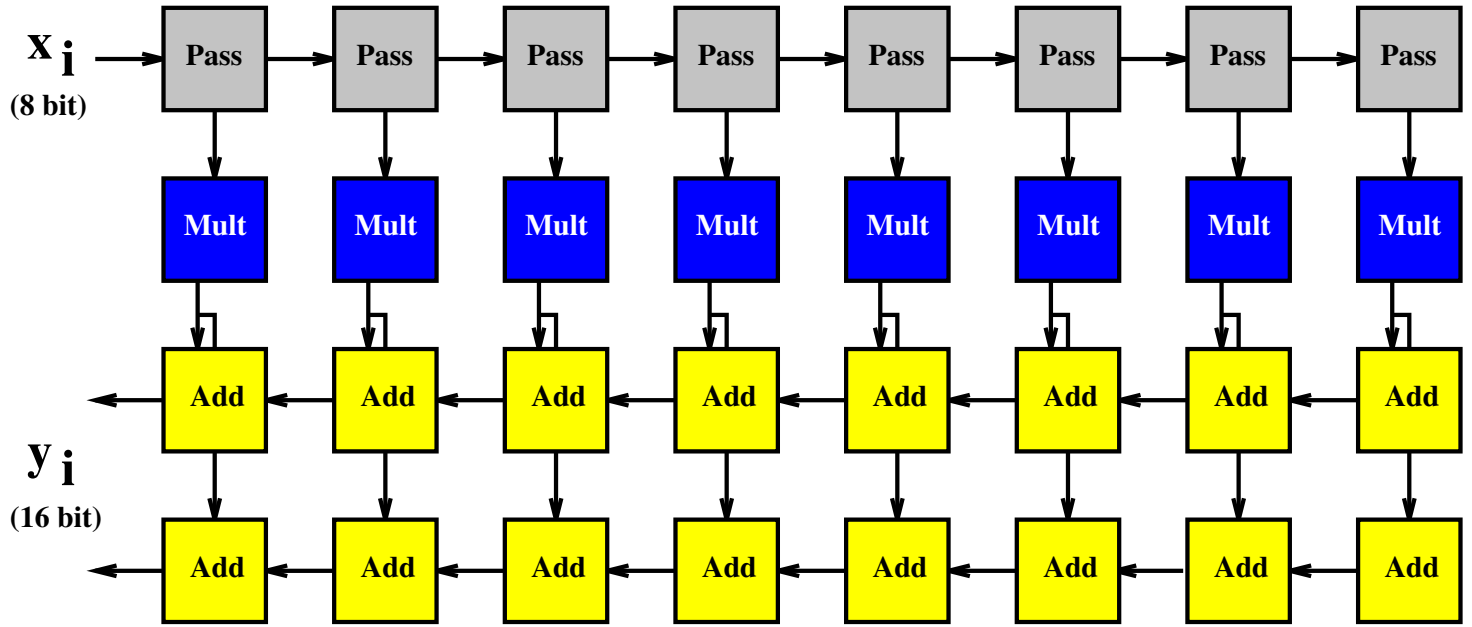
$$y_i = w_1 \cdot x_i + w_2 \cdot x_{i+1} + \cdots + w_k \cdot x_{i+k-1}$$

Examples based on:

- **8-bit samples**
- **16-bit accumulate**

MIT AI Lab

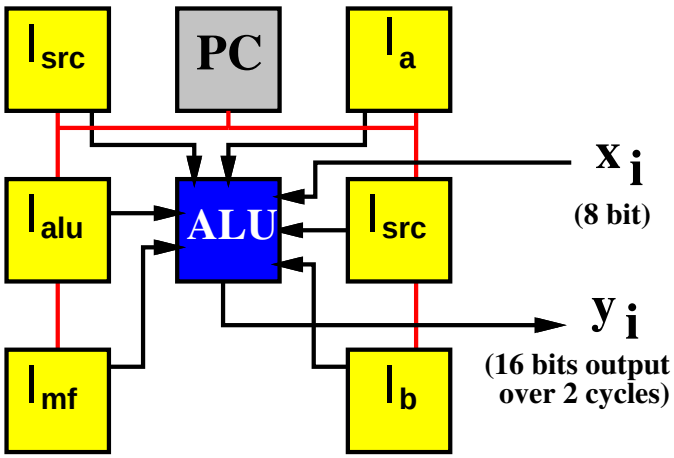
MATRIX FIR – Systolic (Spatial)



Area: $4k$ BFUs as shown ($2k + 4$ in practice)
(k == number of filter weights)

Throughput: 2 cycles per 16-bit result (25MHz)

MATRIX FIR – Microcoded (Space Limited)



Label	ALU Op	PC
inn	Rs ← Rs × Rw	
	Rw ← ×-continue	
	Rl ← Rs + Rl	
	Rh ← Rw +-continue Rh	
ent	Rxp ← Rxp + 1 ; Match $(k+1)$	BNE x2
	Rs ← < Rxp >	(br slot)
	Rxp ← 65	
	Rs ← < Rxp >	
x2	Rwp ← Rwp + 1 ; Match $(k+1)$	BNE inn
	Rw ← < Rwp >	(br slot)

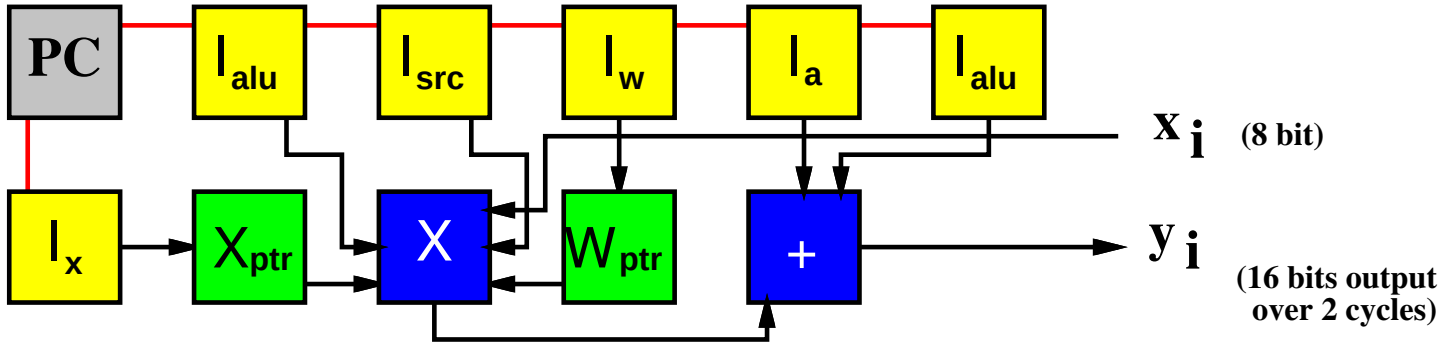
Area: 8 BFUs

Throughput: $(8k+9)$ cycles per 16-bit result

(k == number of filter weights)

MIT AI Lab

MATRIX FIR – VLIW



Label	Xptr unit	Wptr unit	PC	MPY unit	+-unit
innerloop	Xptr++ MOD k | 64 output Xptr	Wptr++; Match k output Wptr	BNE innerloop (pipelined branch slot)	$<$ Xptr $>$ $\times$ $<$ Wptr $>$ $\times$-continue	Rhigh$\leftarrow$ Rhigh + MPY-result Rlow $\leftarrow$ Rlow + MPY-result

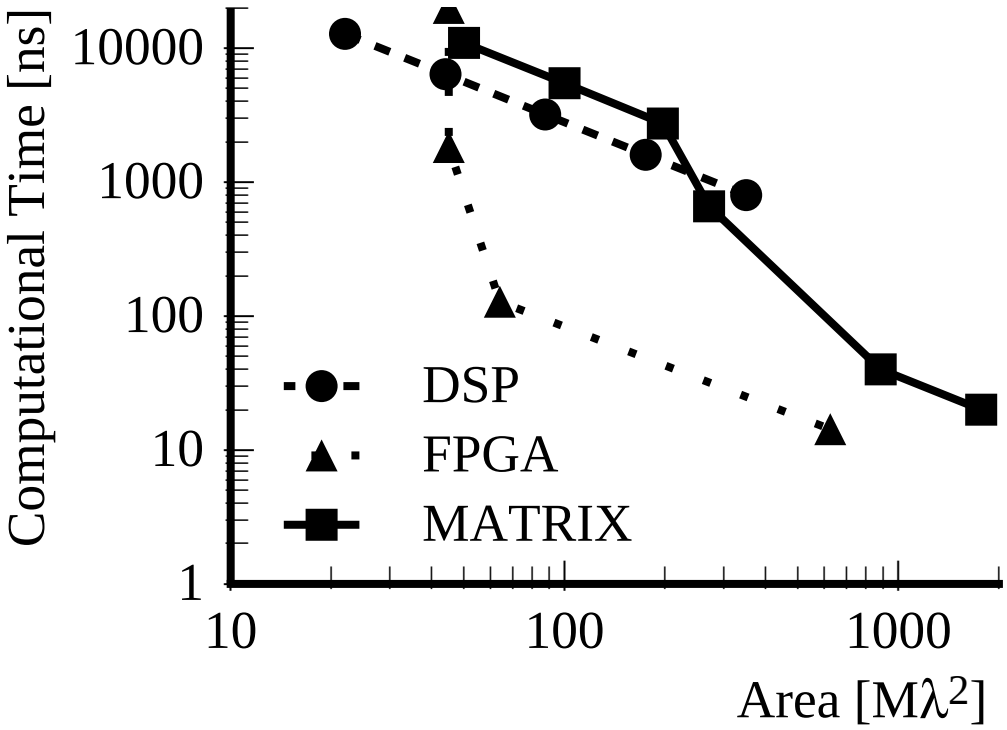
Area: 11 BFUs

Throughput: $(2k + 1)$ cycles per 16-bit result

(k == number of filter weights)

MIT AI Lab

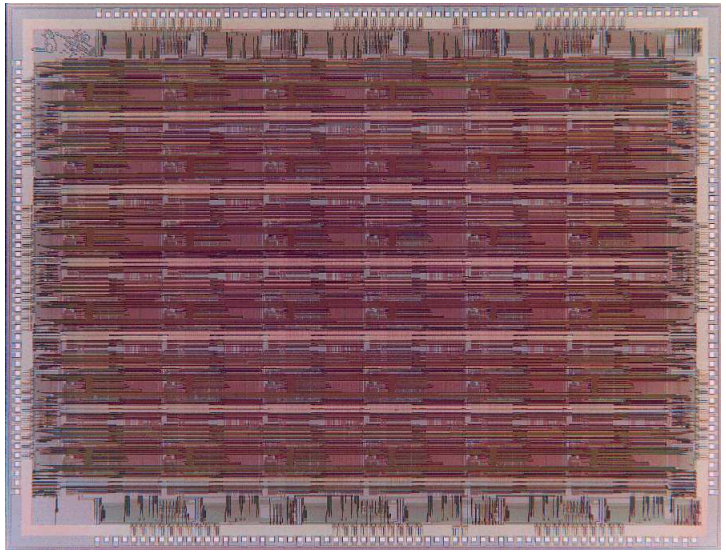
MATRIX Area-Time Comparison



Area-Time for 16 TAP Filter

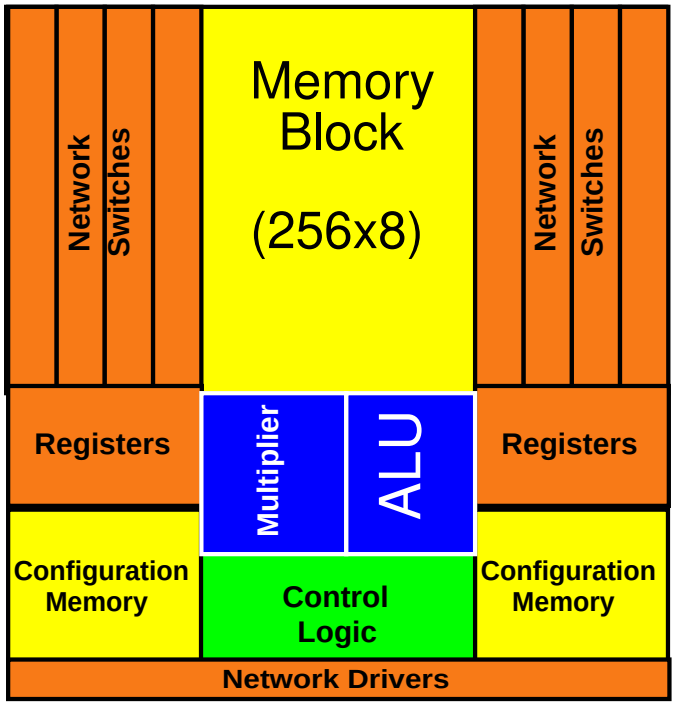
MATRIX Implementation

Technology	0.6μm CMOS	(3 metal)
Die Size	13mm$\times$10mm	
Pin Count	228 (124 I/O)	
Clock Rate	50 MHz	
BFUs	36 (prototype)	(100's in typical devices)
Performance	1.8 Gops/sec	(8-bit Ops)



MATRIX BFU Details

- BFU Size 1.3mm$\times$1.7mm
- Features
  - 8-bit ALU
  - 8$\times$8$\rightarrow$16 Multiplier
  - 256 Byte SRAM (dual port)
  - 8 Byte-wide input ports
  - 28 input sources
  - 11 Byte-wide output drivers
  - Config. Memory: 90 Bytes



Summary

- **Traditional Architecture fix instruction distribution**
  - Large amounts of silicon may sit idle $\Rightarrow$ inefficient
- **MATRIX is a novel architecture which addresses this problem with:**
  - Resource Balance
  - Configurable Instruction Distribution
  - Deployable Resources
- **Resources may be deployed in an application specific manner**
  - Datapaths, Memory, Instruction Distribution

MIT AI Lab

Reinventing Computing (MIT)

<http://www.ai.mit.edu/projects/transit/rc_home_page.html>

MATRIX Documents

<http://www.ai.mit.edu/projects/transit/matrix_documents.html>

BRASS Project UC Berkeley

<http://www.cs.berkeley.edu/projects/brass/>

MIT AI Lab