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.

Outline

- Problem Identification
- Benefits of Configurable Instruction Distribution
- MATRIX Architecture
- Usage Example – Convolution
- Implementation Details
- Summary

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)

Traditional Approaches

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

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

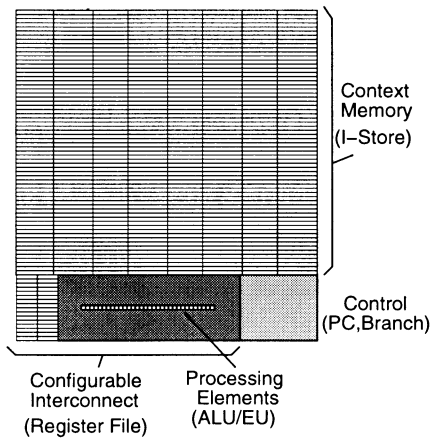
Processors

Dedicate significant area to instruction and data memory

⇒ efficient for heavily multiplexed operation

⇒ inefficient on regular tasks

⇒ lower yielded capacity on fine-grained data



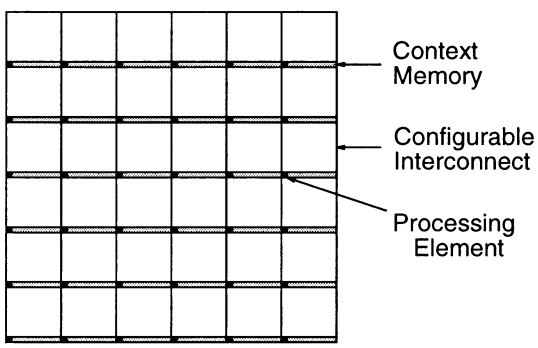
Field-Programmable Gate Arrays

Dedicate most area to fine-grained interconnect

⇒ efficient on very regular operations

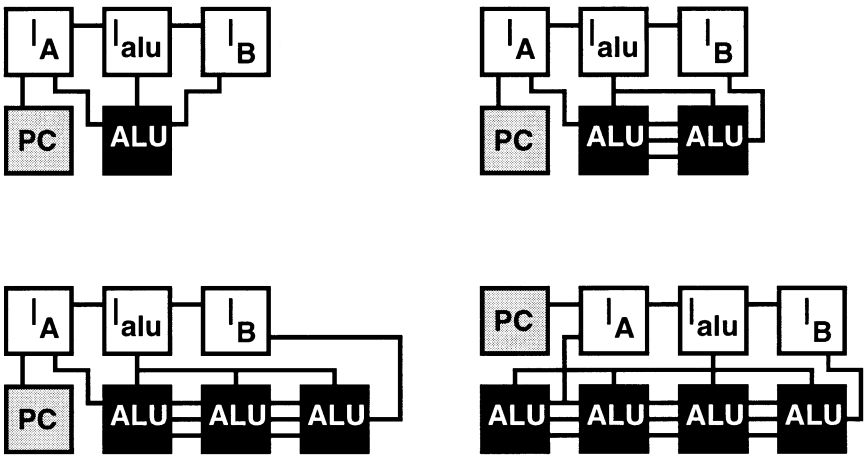
⇒ high fine-grain yield

⇒ low yielded capacity on irregular tasks



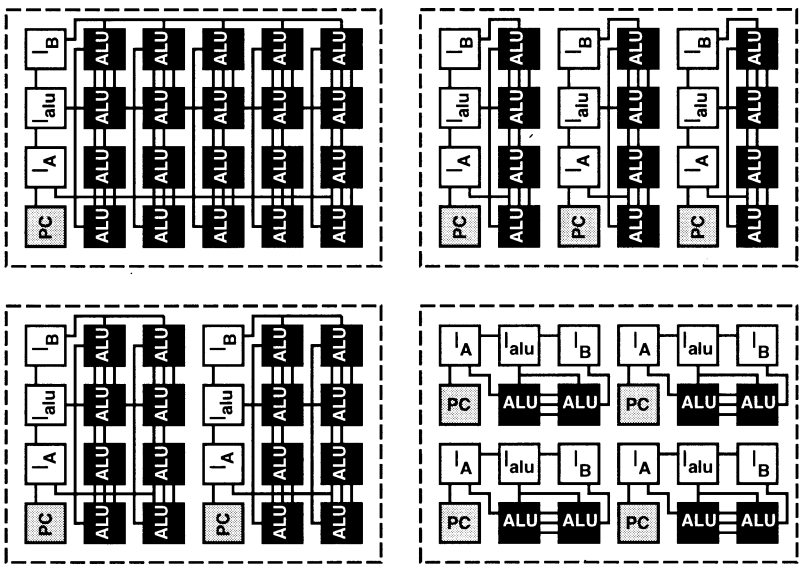
MATRIX Benefits: Configurable Granularity

Select granularity according to needs of the application.

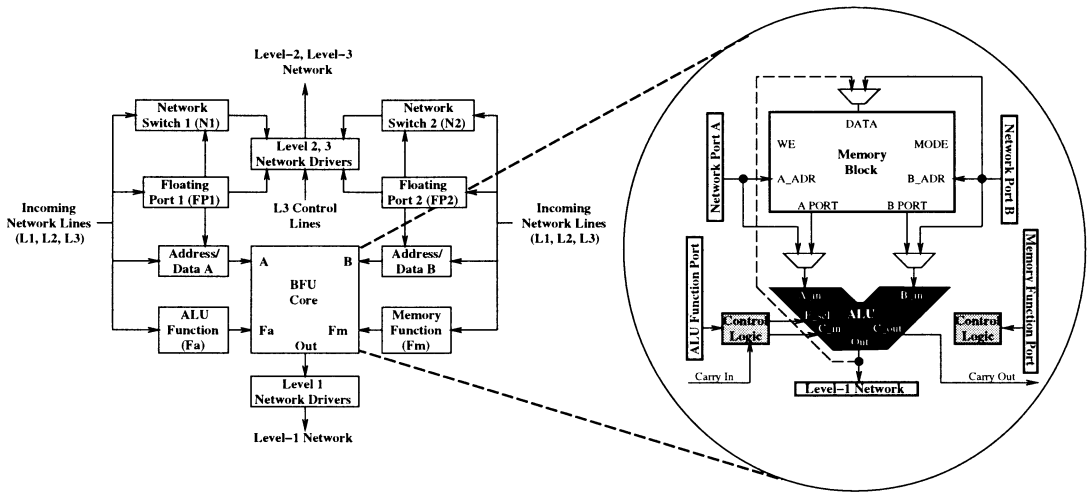


MATRIX Benefits: Configurable Instruction Distribution

Just as much control granularity as the problem needs



MATRIX Architecture
Basic Functional Unit (BFU)

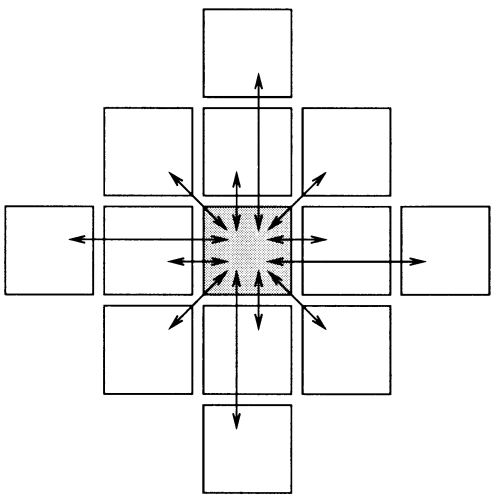


- Same network servers data, instructions, and addresses
- Can use BFU as Instruction Store, Data Memory, or Compute Unit

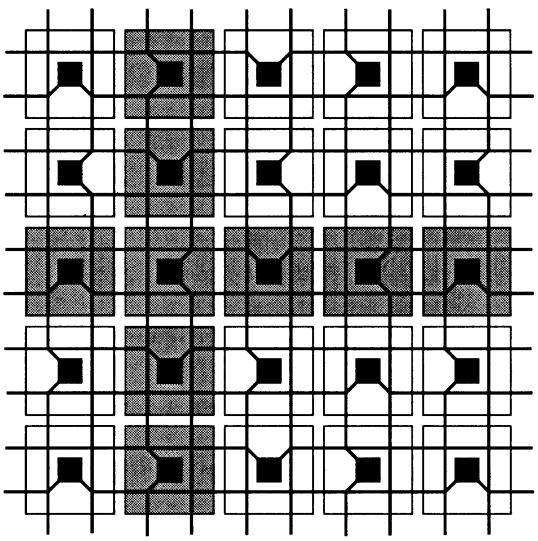
MIT AI Lab

MATRIX Architecture (Network)

Nearest Neighbor Interconnect



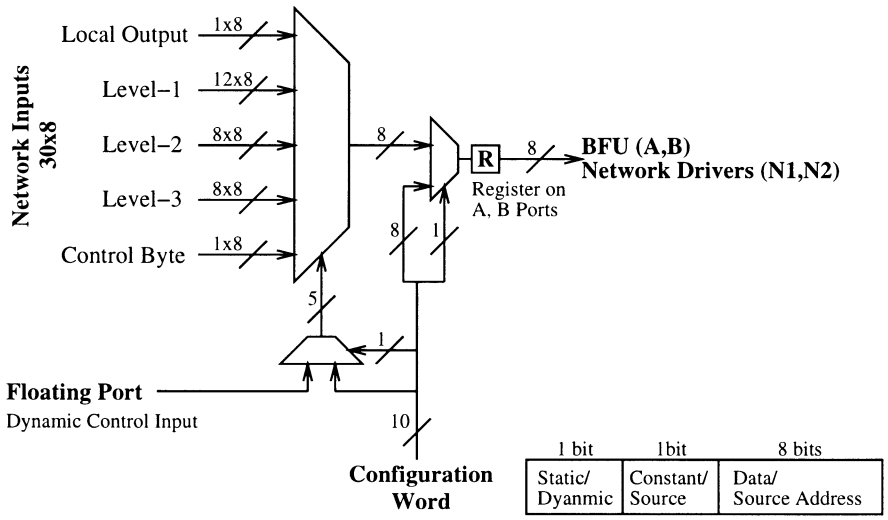
Length Four Bypass Interconnect



Global - 4 Lines per Row/Column

MIT AI Lab

MATRIX Port Architecture



MIT AI Lab

Usage Example
Finite Impulse Response Computation

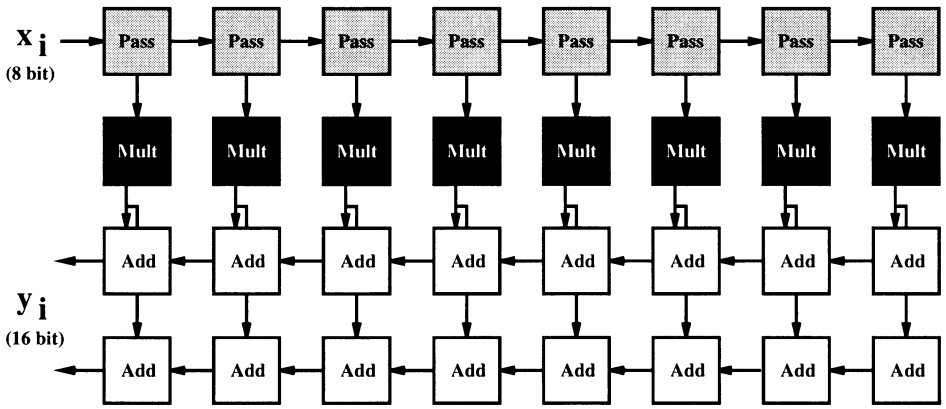
$$y_i = w_1 \cdot x_i + w_2 \cdot x_{i+1} + \cdots + w_k \cdot x_{i+k-1}$$

Examples based on:

- 8-bit samples
- 16-bit accumulate

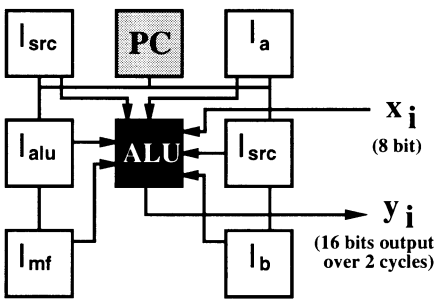
MIT AI Lab

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)

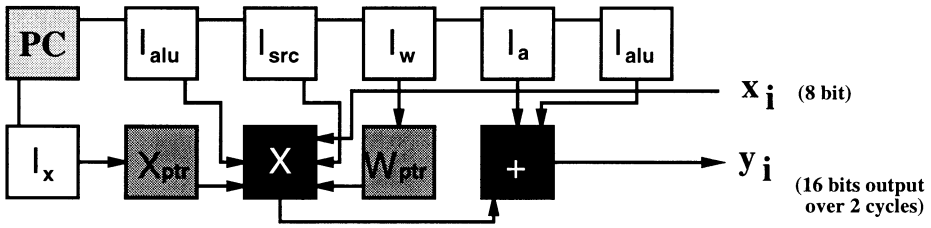
MATRIX FIR – Microcoded (Space Limited)



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)

MATRIX FIR – VLIW



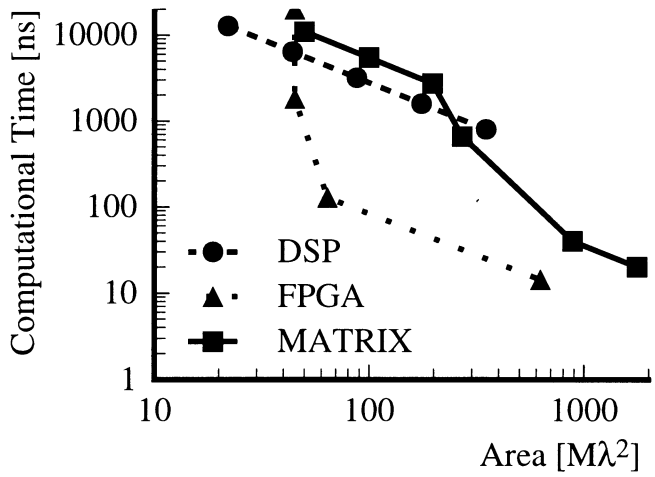
Label	Xptr unit	Wptr unit	PC	MPY unit	+-unit
innerloop	Xptr++ MOD k | 64 output Xptr	Wptr++; Match k output Wptr	BNE innerloop (pipelined branch slot)	< Xptr > × < Wptr > ×-continue	Rhigh← Rhigh + MPY-result Rlow ← Rlow + MPY-result

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

MIT AI Lab

Summary

- **Traditional Architecture fix instruction distribution**
  - Large amounts of silicon may sit idle ⇒ 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

Reinventing Computing

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

MATRIX Documents

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

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)
(envision 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**