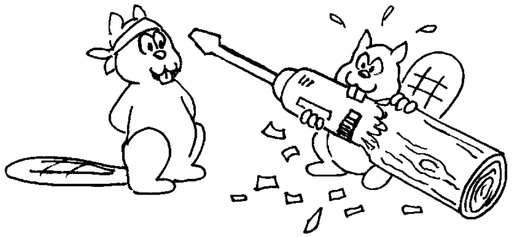
MATRIX
A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources

Ethan Mirsky <eamirsky@ai.mit.edu>

André DeHon <andre@mit.edu>

Reinventing Computing



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

- Motivation
- Basic MATRIX Architecture
- MATRIX Port Architecture
- Prototype Implementation
- Application Sample – Convolution
- Placement in General-Purpose Computing World
- Summary

Motivation

Issues for General-Purpose Computing Architectures:

- Granularity (datapath width)
- Instruction Memory and Distribution Area
  - Amount of instruction memory
  - Number of instruction streams
  - Coupling of instruction streams
  - Composition of instruction streams

"Traditional" Architectures

Processors – coarse grain, dedicated Imem/control space
- → low computational density
- → lower yielded capacity on fine-grained data

FPGAs – fine grain, minimal control/Imem resources
- → high raw computational density
- → fine-grain yield
- → low yielded capacity on irregular tasks

DPGAs – fine granularity, multiple instructions/element
- → slightly reduction in computational density
- → fine-grain yield
- → more robust yielded capacity as operation regularity decreases

“Traditional” Architectures

Processors – coarse grain, dedicated Imem/control space
- → low computational density
- → lower yielded capacity on fine-grained data

FPGAs – fine grain, minimal control/Imem resources
- → high raw computational density
- → fine-grain yield
- → low yielded capacity on irregular tasks

Motivation

All Modern Architectures

- Fix instructions/element ratio at fabrication time
- Fix granularity at fabrication time
- Dedicate control paths/resource at fabrication time

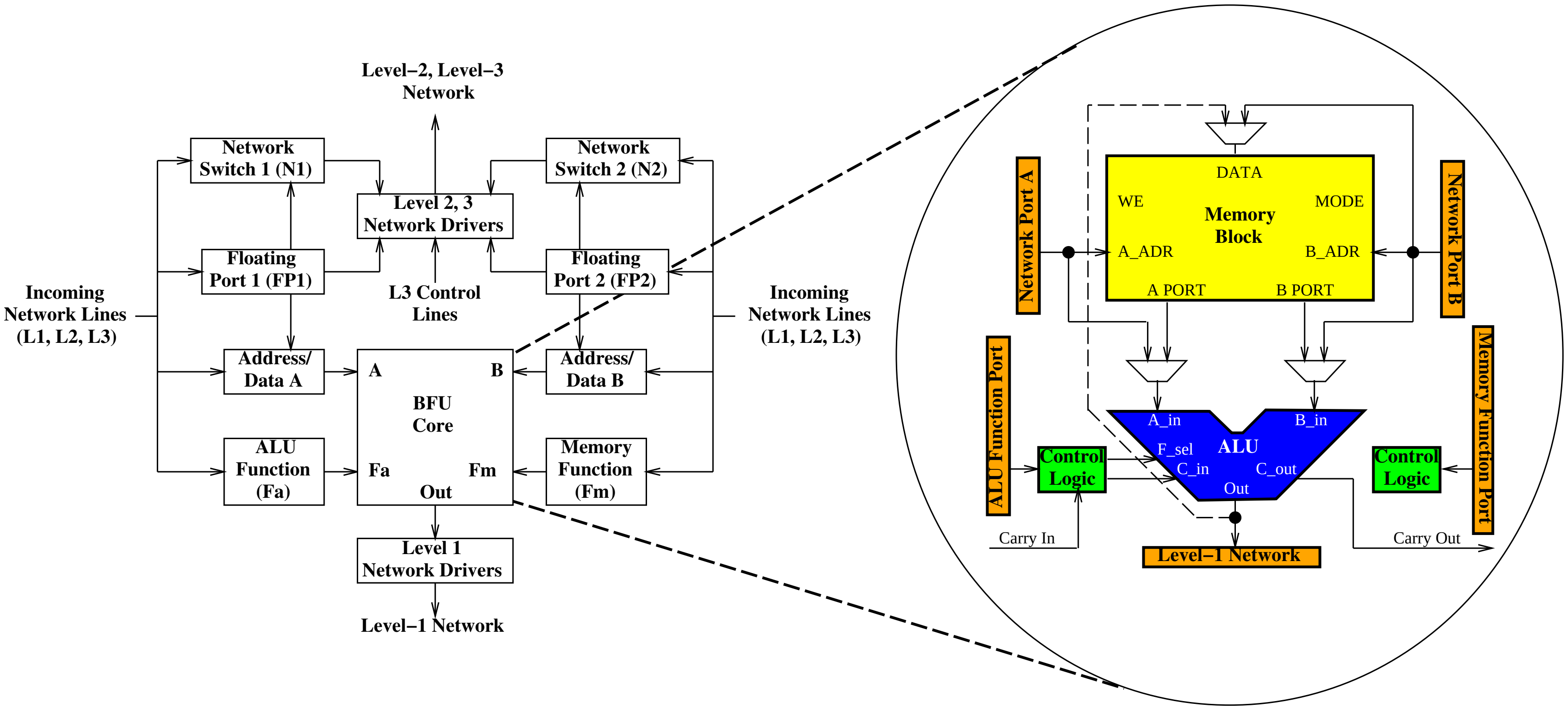
Decisions differentiate conventional architectures

MATRIX

- Unifies instruction and datapath resources
- Defers many of the decisions until after fabrication

MATRIX Architecture

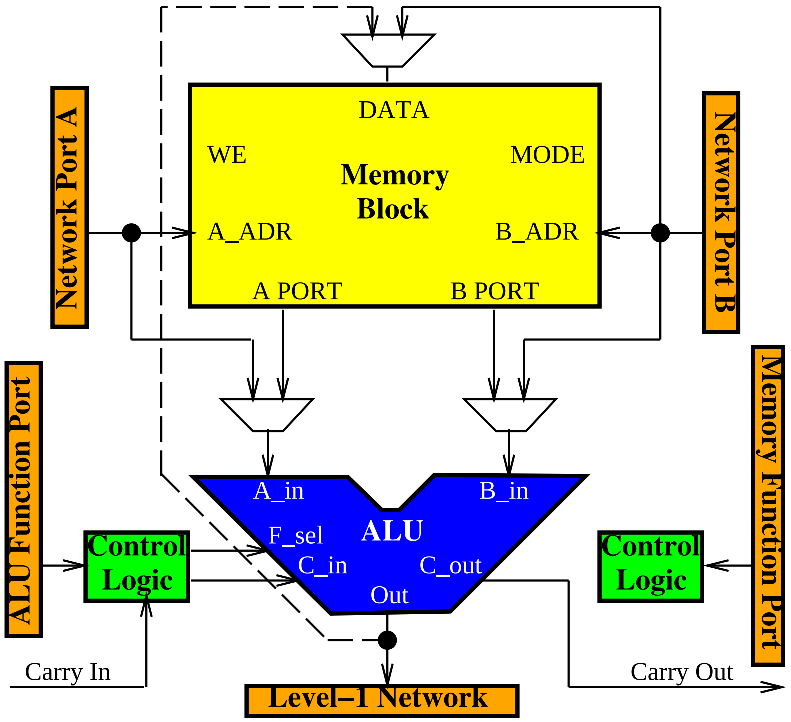
MATRIX Architecture (BFU)



MIT AI Lab

MATRIX Architecture (BFU)

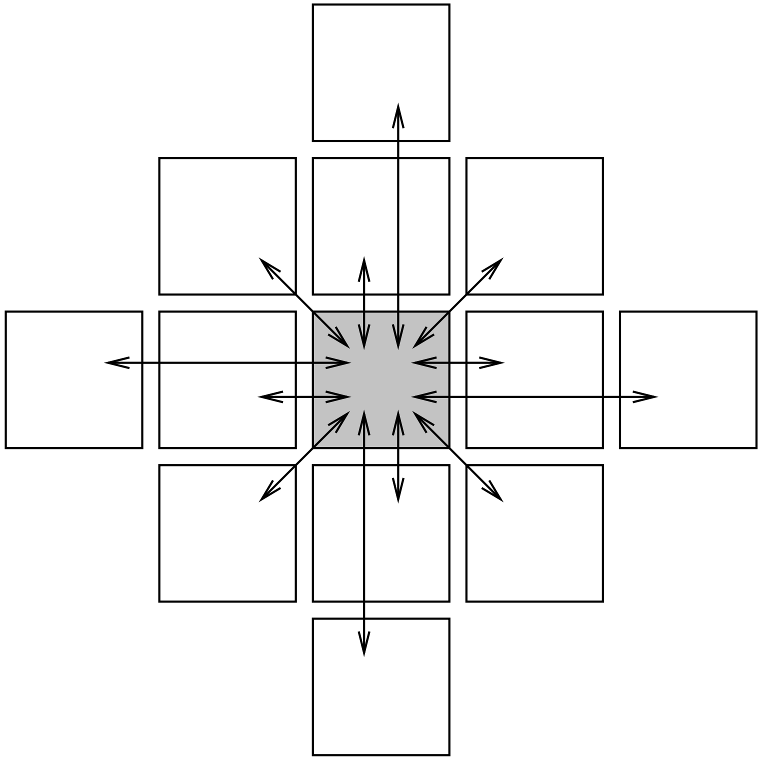
Memory	**256×8**
(or)	**128×8 RF**
	2R, 1W
ALU	**8-bit**
(incl.)	**8×8 multiply**
Control	pattern match
	20×8 NOR
	reduce



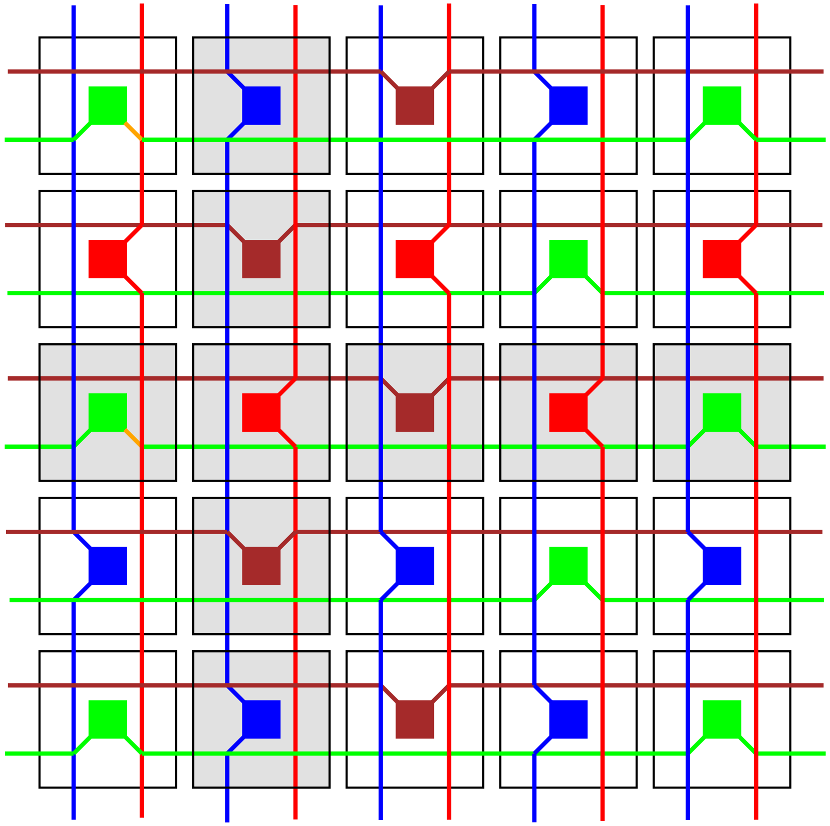
MIT AI Lab

MATRIX Architecture (Network)

Nearest Neighbor Interconnect



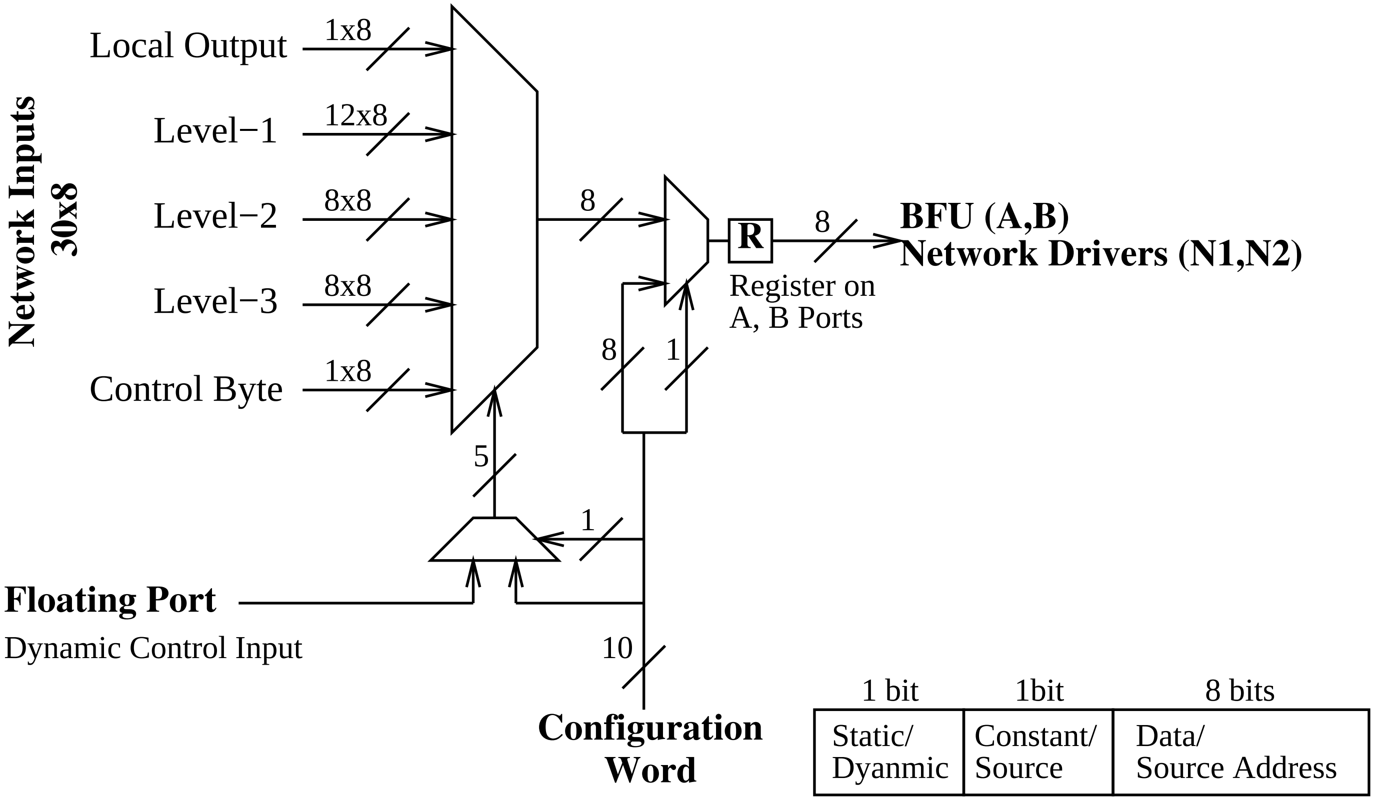
Length Four Bypass Interconnect



Global – 4 Lines per Row/Column

Port Architecture

Dynamic Port Architecuture

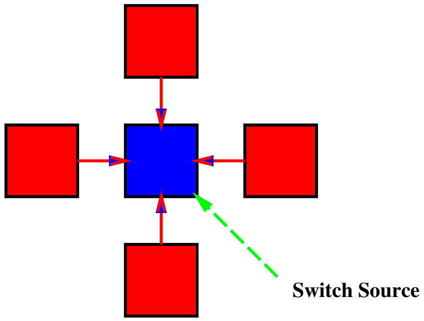


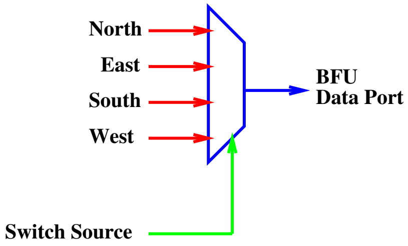
MIT AI Lab

Dynamic Port Usage Example

Guassian Convolution

$$NewVal = \frac{4 \cdot Val + Val_N + Val_E + Val_S + Val_W}{8}$$



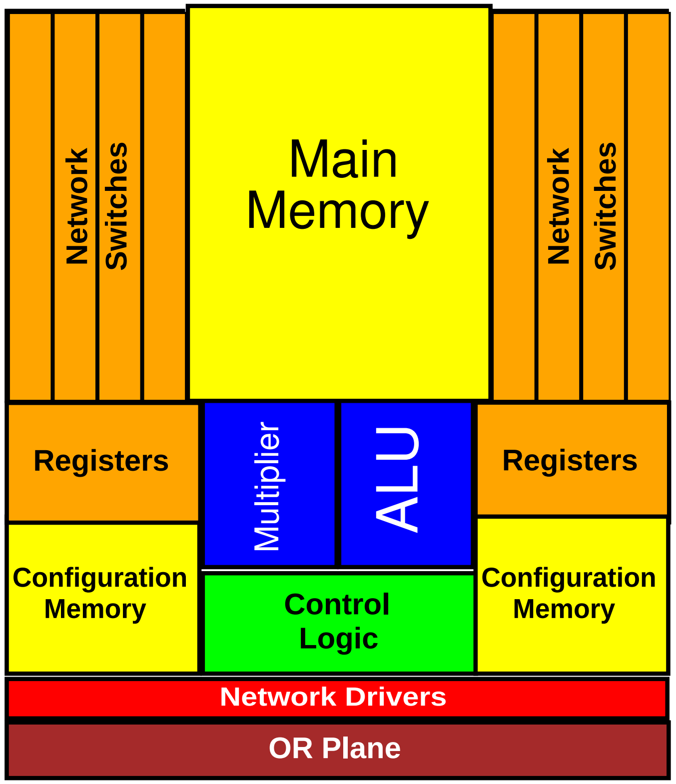


ALU Op	Switch Source
NewVal ← 4 * OldVal	–
NewVal ← NewVal + PortVal	North
NewVal ← NewVal + PortVal	East
NewVal ← NewVal + PortVal	South
NewVal ← NewVal + PortVal	West
NewVal ← RightShift(NewVal)	–
NewVal ← RightShift(NewVal)	–
NewVal ← RightShift(NewVal)	–

MATRIX Implementation

MATRIX Implementation

Technology	0.5μ CMOS
BFU Size	$1.2\text{mm}\times 1.5\text{mm}$
	$(\approx 29\text{M}\lambda^2)$
Data Width	8-bit
Memory	256×8
Cycle	10 ns (univ.)



Network	50%
Memory	30%
Control	12%
ALU & Multiplier	8%

Extended Example:
Finite-Impulse Response

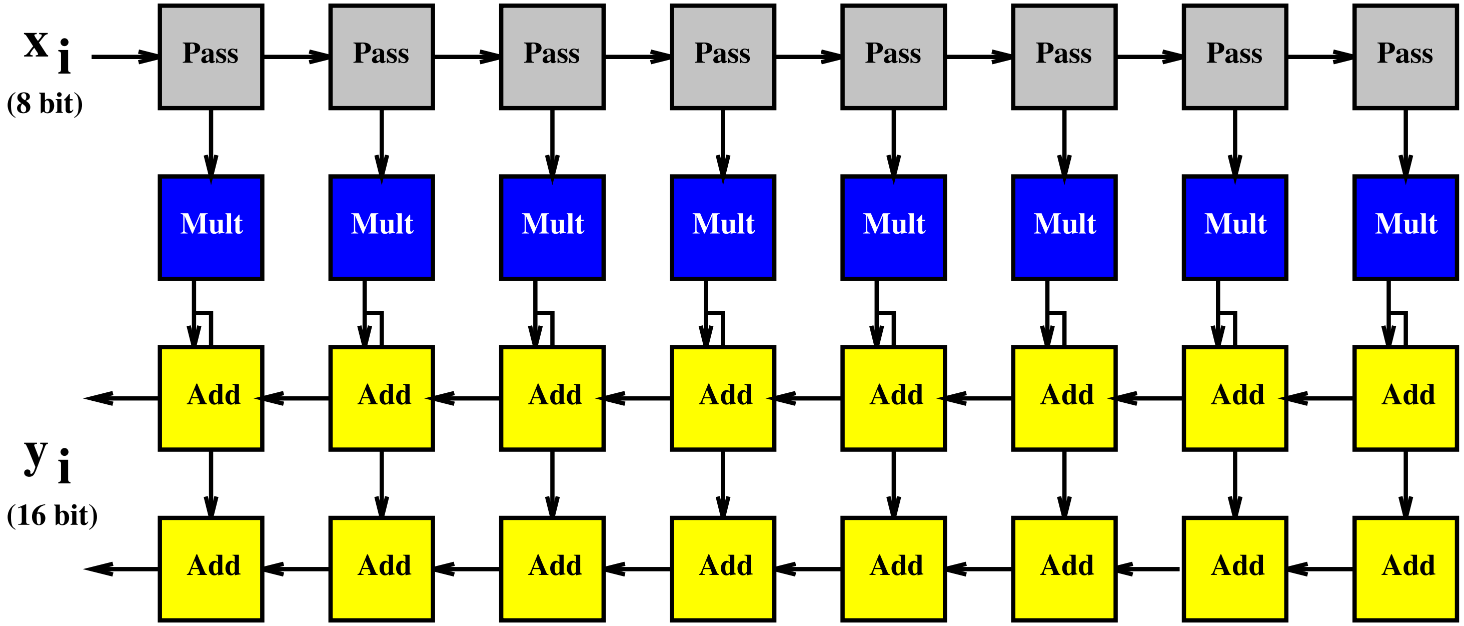
Finite Impulse Response Computation

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

Examples based on:

- 8-bit samples
- 16-bit accumulate

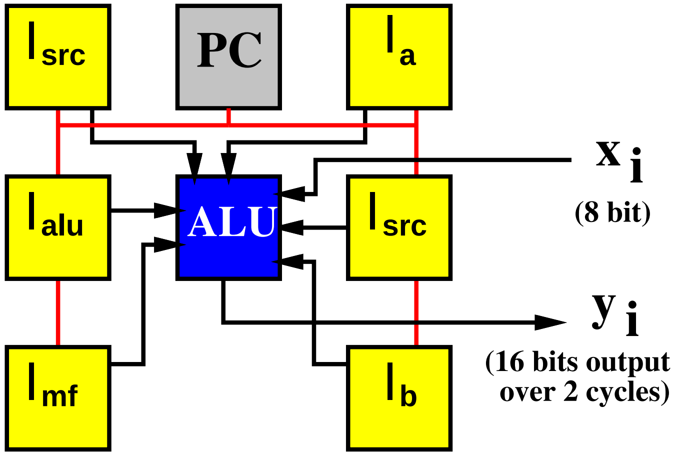
MATRIX FIR – Systolic (Spatial)



In practice can do in half this space at 50 MHz

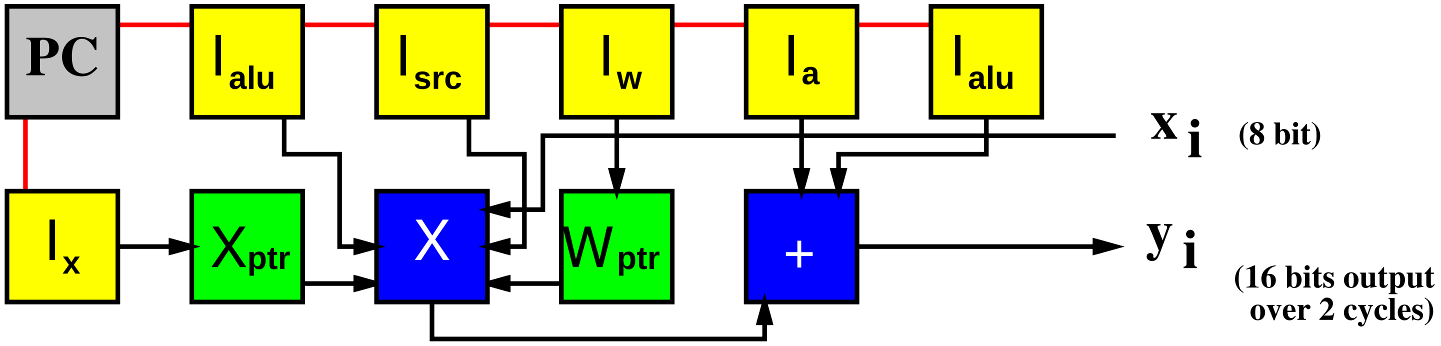
(use level-2 net for pipelining and retiming, byte-serial adds)

MATRIX FIR – Microcoded (Space Limited)



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

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

FIR Survey – 8×8 multiply, 16-bit Accumulate

Architecture	Reference	$\frac{\text{Filter TAPs}}{\lambda^2 \cdot s}$
16b DSP	ISSCC86	0.090
	CICC92	0.072
32b RISC MSTEP	MIPS-X	0.029
32b RISC/DSP	VSP8	0.022
64b RISC	1996 Alpha	0.064
systolic		0.87
MATRIX microcode		0.054
VLIW		0.16
XC4K	App. Note	1.0†
	ICSPAT93	0.080
PADDI2	ISSCC95	0.93
Full Custom	JSSC89	6.1‡
	JSSC90	3.5

† – symmetric filter; ‡ – 24-bit accum.

MIT AI Lab

Deployable Resources

- Programmable Systolic (FPGA)
  - Operators require space
  - Lower bound on implementation size
- Fully Microcoded (Processor)
  - Individual resources perform multiple operations
  - Requires overhead for instructions
- MATRIX
  - Pay overhead only when needed
  - Performs both styles efficiently

Relationship to Conventional Architectures

Instruction Taxonomy – Conventional

Control Threads (PCs)	Instructions	Architecture/Examples
0	n/a	Hardwired Functional Unit Group (*e.g.* ECC/EDC unit, FP MPY, hardware systolic)
	n	FPGA, Programmable Systolic Array
1	1	SIMD
	n	VLIW, PADDI, DPGA
n	n	MIMD (traditional), PADDI-2

MATRIX Instruction Control

Conventional Architectures

Efficiency depends on **match** between

- Application characteristics
- Architecture characteristics

MATRIX – Delayed binding

- Instructions
- Control

→ Robust on larger range of application requirements

Summary

Summary (1 of 2)

- Introduced a novel, reconfigurable architecture
  - Intermediate-grain (8-bit)
  - Multiple levels of configuration
  - Unified
    * instruction control
    * datapath
    * memory resources
- Architecture allows deployment of resources in an application-specific manner.

Summary (2 of 2)

- Architecture achieves
  - Higher raw computational density than processors
  - More flexible control than other configurable architectures

    →Robust yield of available capacity

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