Tutorial HotChips 01

Silicon Architectures for Wireless Systems - Part 2 Configurable Processors

Jan M. Rabaey

BWRC

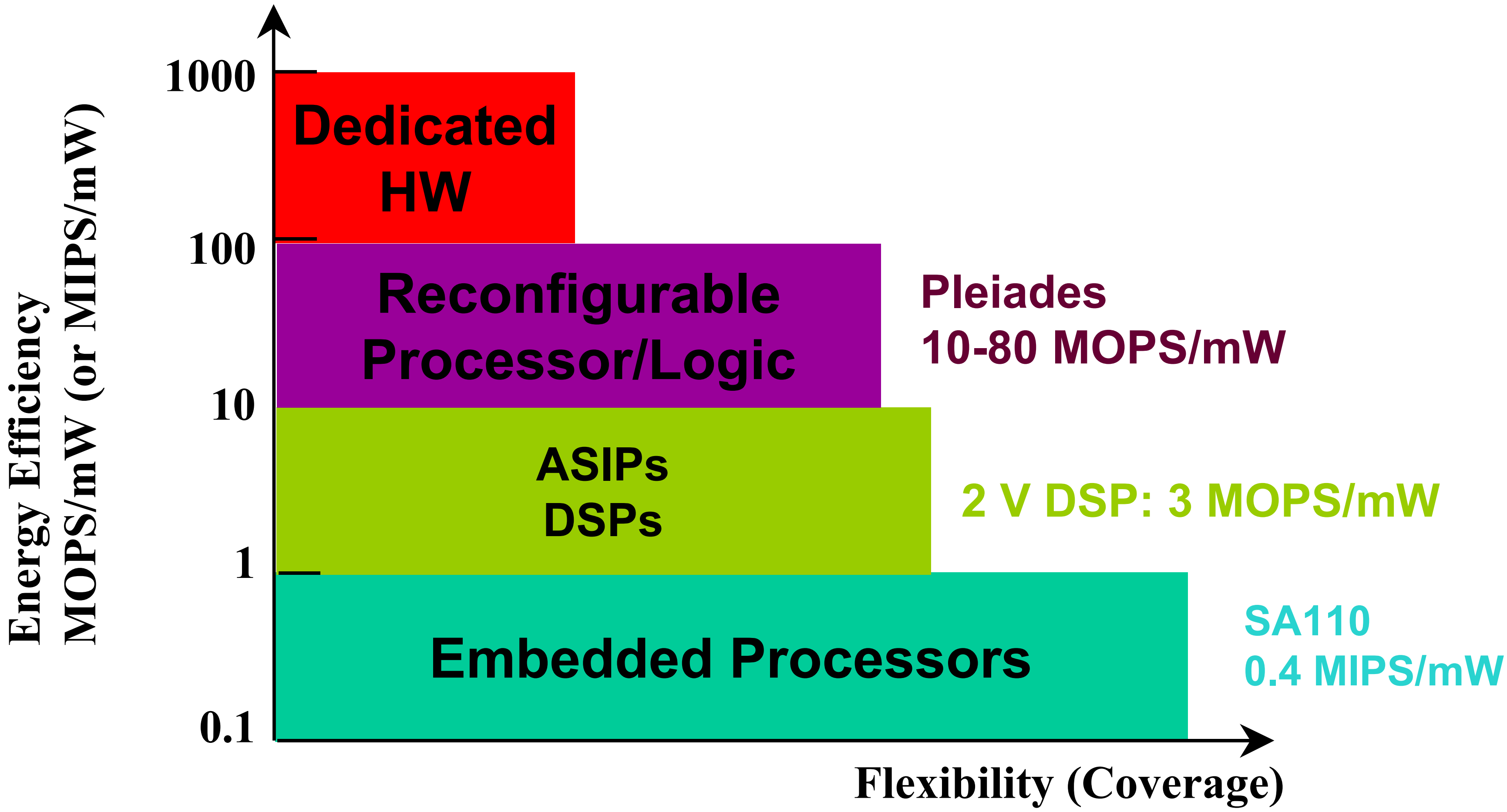
University of California @ Berkeley

http://www.eecs.berkeley.edu/~jan

With contributions from J. Wawrzynek and A. Dehon

Berkeley Wireless
Research Center

The Energy-Flexibility Gap



Pleiades
10-80 MOPS/mW

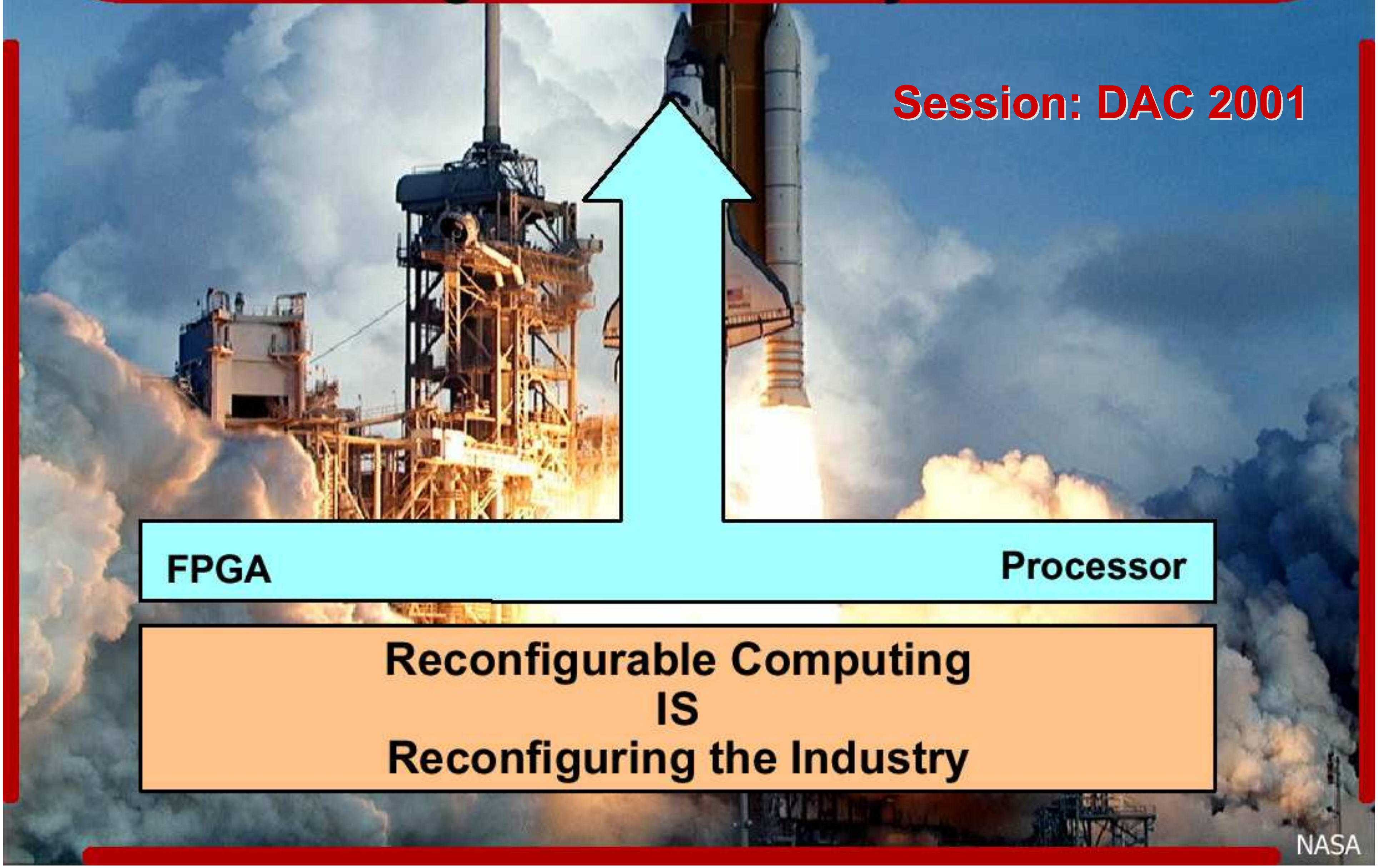
2 V DSP: 3 MOPS/mW

SA110
0.4 MIPS/mW

Berkeley Wireless
Research Center

Reconfigurable Industry Take-off

Session: DAC 2001



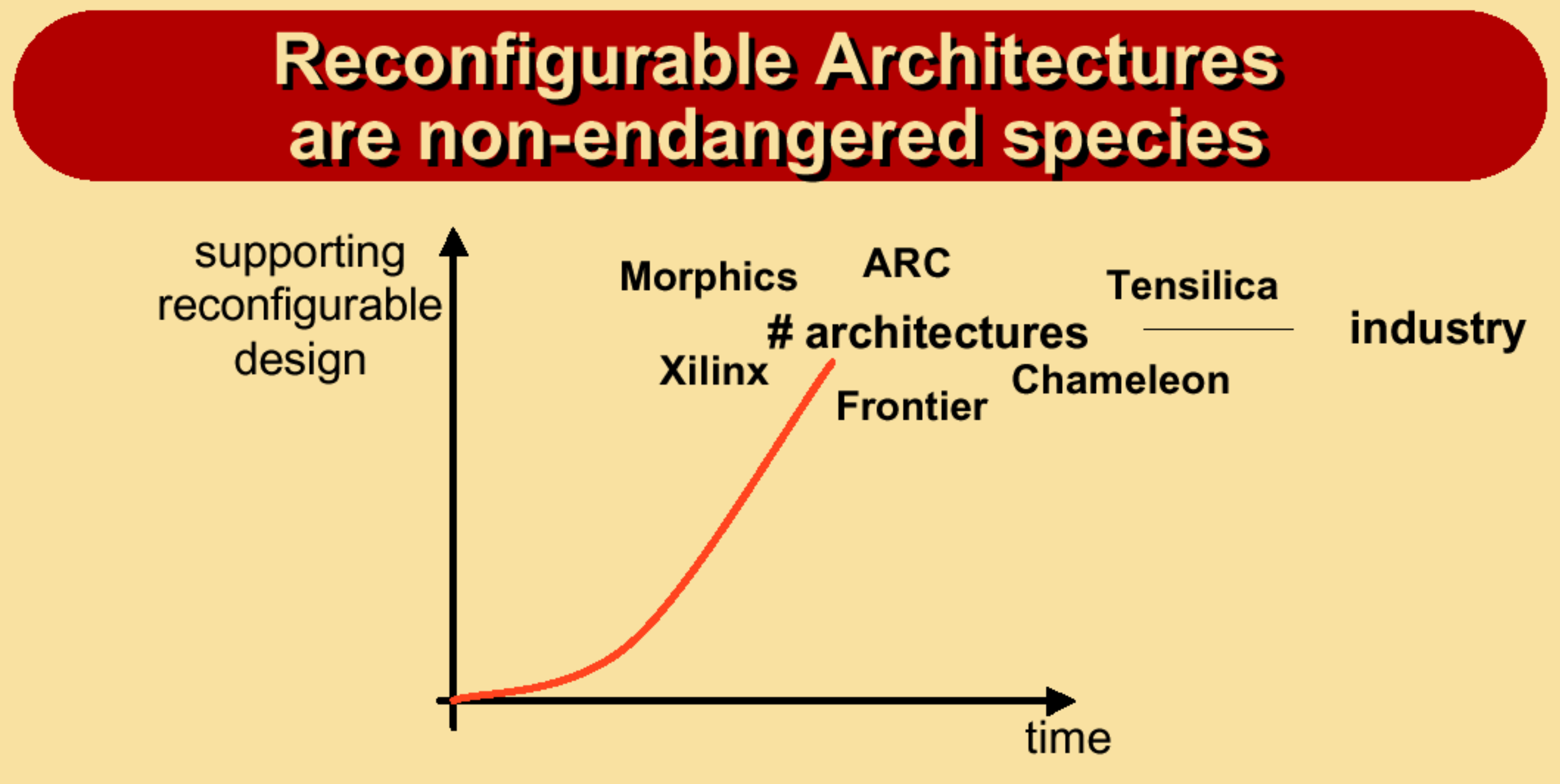
FPGA

Processor

Reconfigurable Computing
IS
Reconfiguring the Industry

NASA

The Growth of Reconfigurable



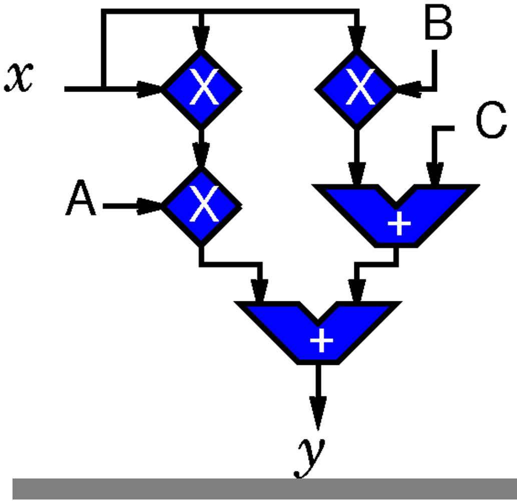
Reconfigurable Architectures are non-endangered species

Source: Schaumont et al., DAC 2001

(Re)configurable Computing: Merging Efficiency and Versatility

Spatially programmed connection of processing elements.

$$y = Ax^2 + Bx + C$$



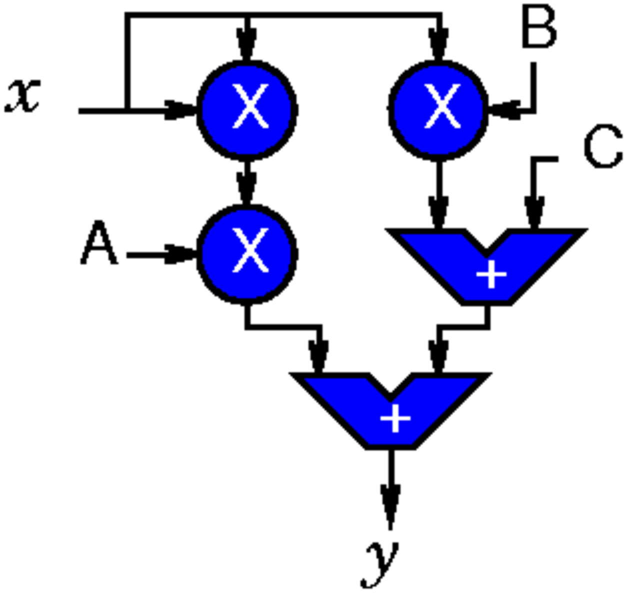
"Hardware" customized to specifics of problem.

Direct map of problem specific dataflow, control.

Circuits "adapted" as problem requirements change.

Spatial vs. Temporal Computing

Spatial



Temporal

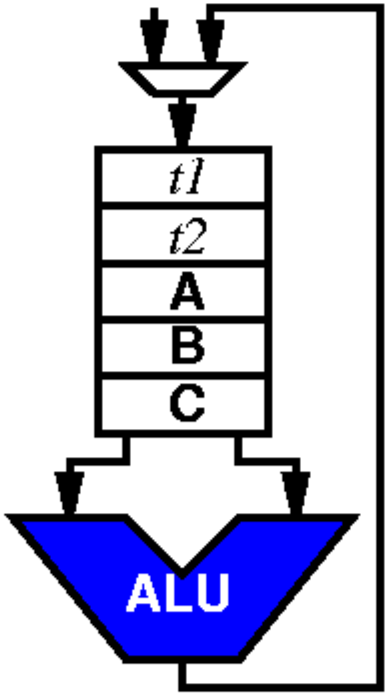
$t1 \leftarrow x$

$t2 \leftarrow \mathbf{A} \times t1$

$t2 \leftarrow t2 + \mathbf{B}$

$t2 \leftarrow \mathbf{t2} \times t1$

$y \leftarrow \mathbf{t2} + \mathbf{C}$



Source: A. Dehon and J. Wawrzynek

Benefits of Programmable

- Non-permanent customization and application development after fabrication
  - "Late Binding"
- economies of scale (amortize large, fixed design costs)
- time-to-market (evolving requirements and standards, new ideas)

Disadvantages

- Efficiency penalty (area, performance, power)
- Correctness Verification

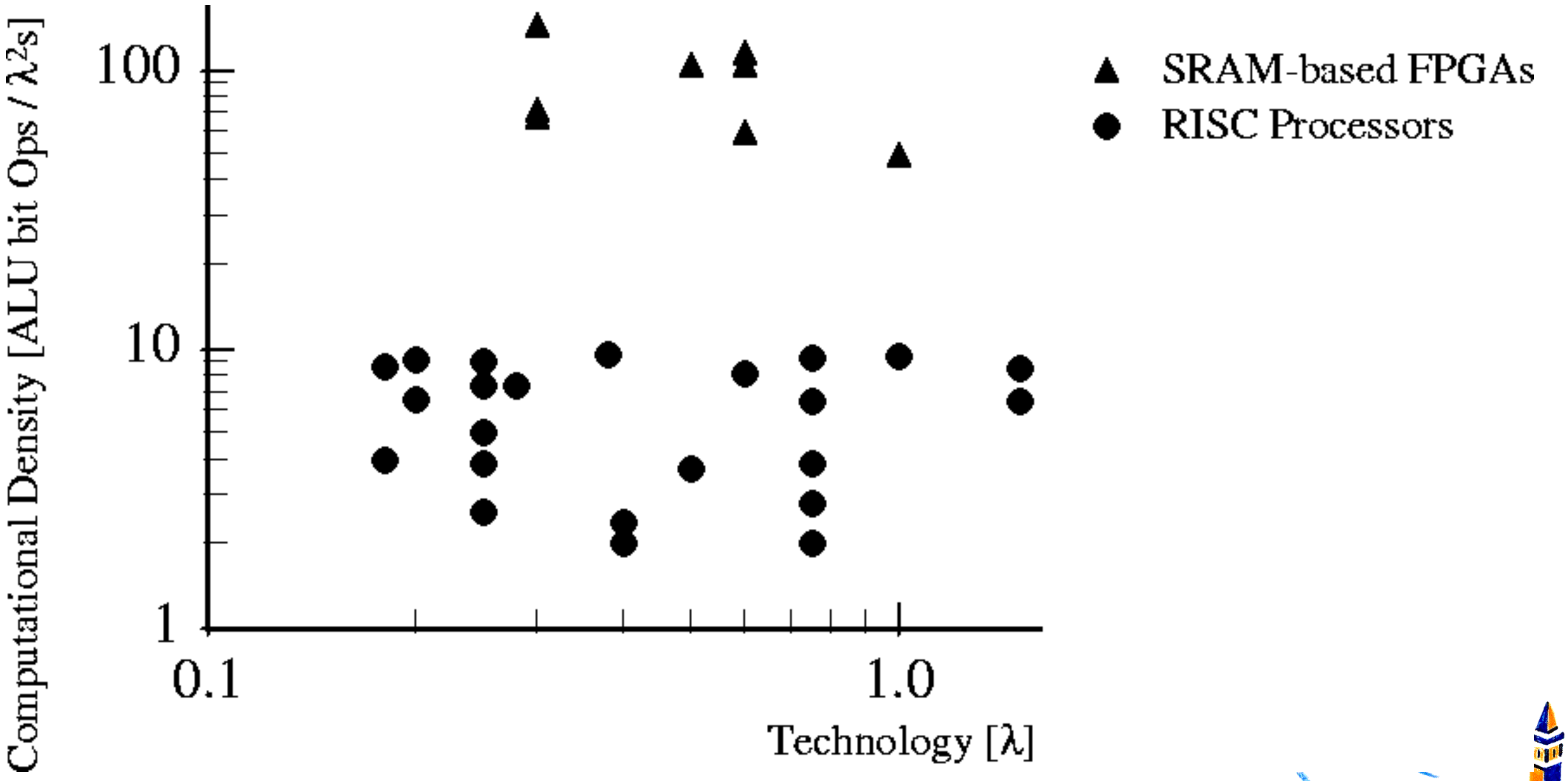
Spatial/Configurable Benefits

- **10x raw density advantage over processors (and increasing)**
- **Energy efficiency (potentially)**
- **Locality, regularity, and predictability**
- **Ultimate distributed architecture**
- **Scalable with technology**
  - Relies mostly on increase in computational density
  - Avoids most of the physics pitfalls threatening high-performance computing

Spatial/Configurable Drawbacks

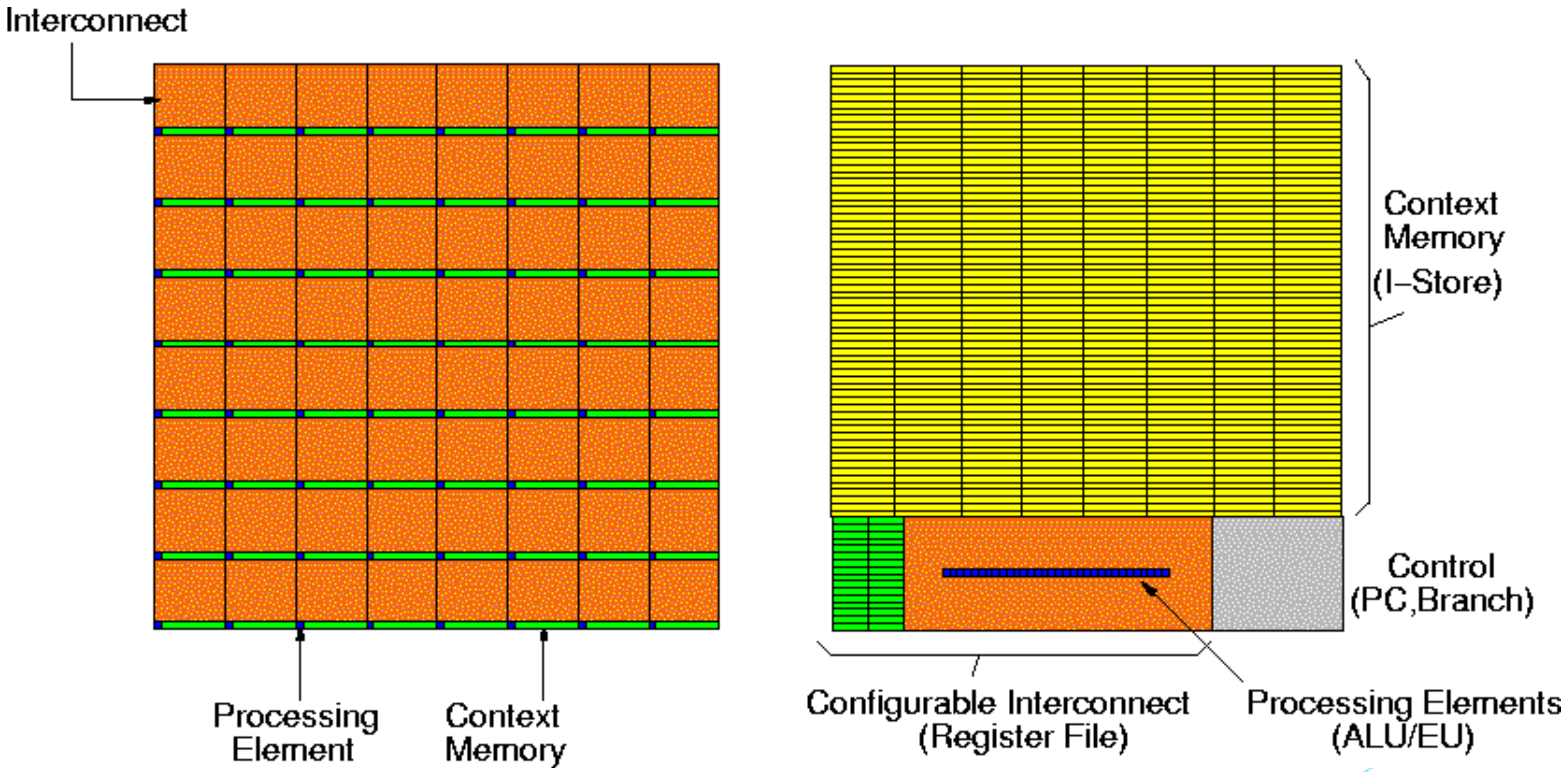
- **Resource management**
  - Each compute/interconnect resource dedicated to single function
  - Must dedicate resources for every computational subtask
  - Infrequently needed portions of a computation sit idle --> inefficient use of resources
  - But … not a real issue when transistors are abundant
- **Potential mismatch between operations and operators**
- **Interconnect plays dominant role**

Density Comparison

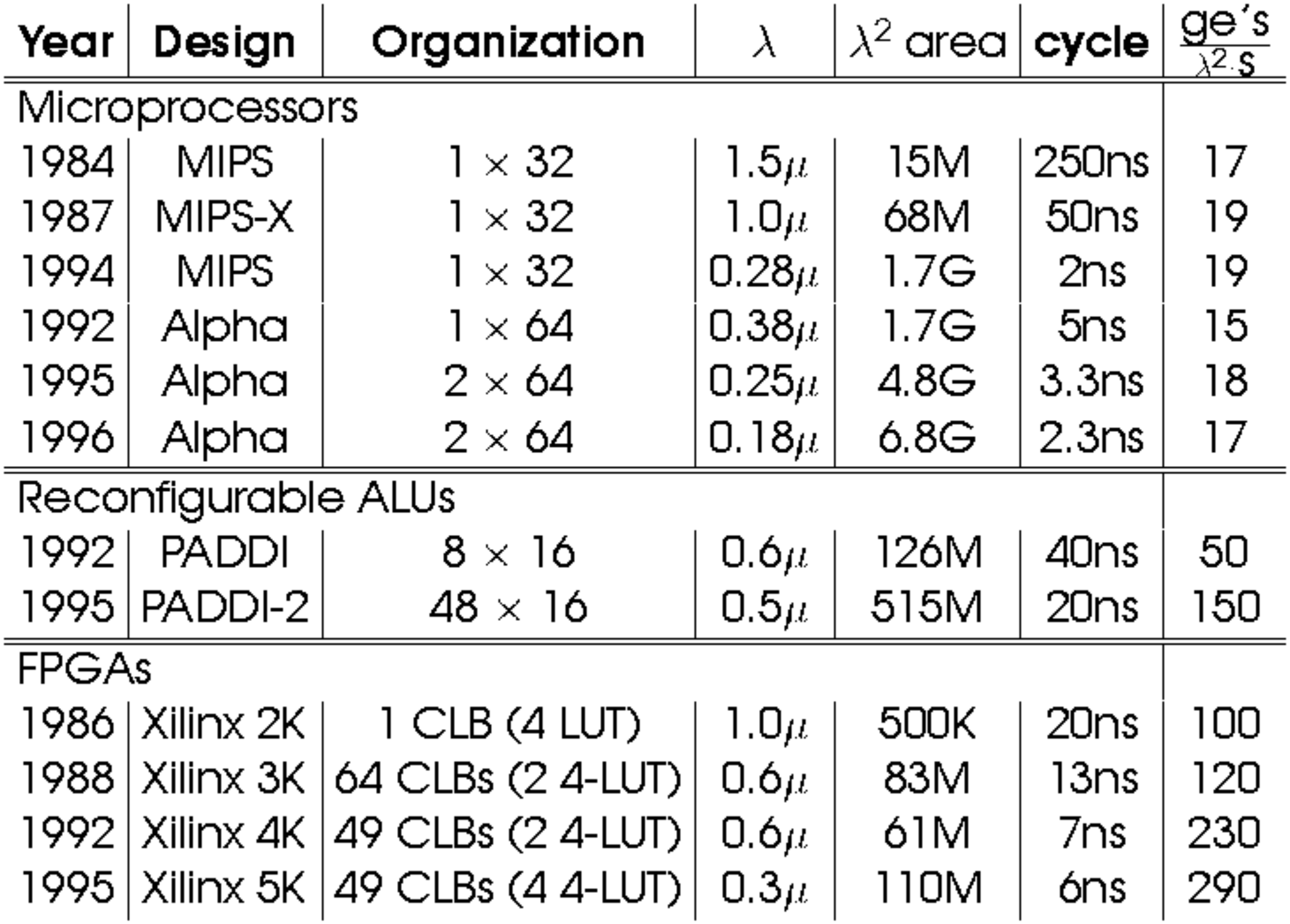


Berkeley Wireless Research Center

Processor vs. FPGA Area



Processors and FPGAs



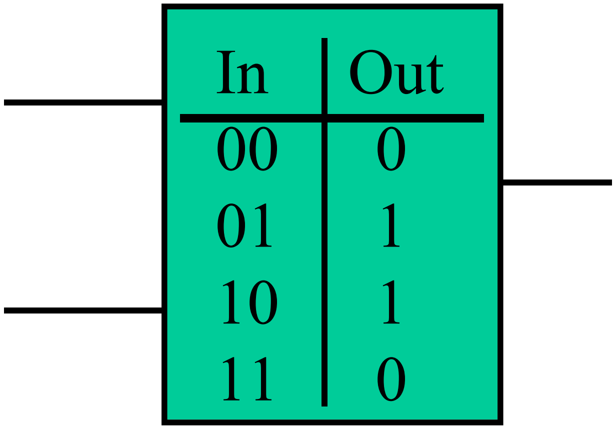
Year	Design	Organization	λ	λ^2 area	cycle	$\frac{ge's}{\lambda^2 \cdot s}$
Microprocessors						
1984	MIPS	1 × 32	1.5μ	15M	250ns	17
1987	MIPS-X	1 × 32	1.0μ	68M	50ns	19
1994	MIPS	1 × 32	0.28μ	1.7G	2ns	19
1992	Alpha	1 × 64	0.38μ	1.7G	5ns	15
1995	Alpha	2 × 64	0.25μ	4.8G	3.3ns	18
1996	Alpha	2 × 64	0.18μ	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

Issues in Configurable Design

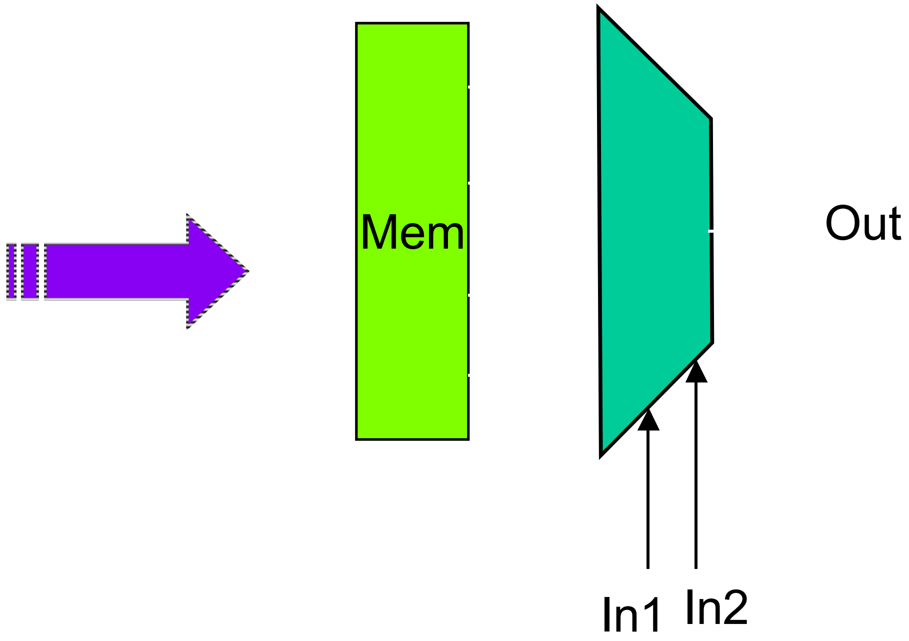
- **Choice and Granularity of Computational Elements**
- **Choice and Granularity of Interconnect Network**
- **(Re)configuration Time and Rate**
  - Fabrication time --> Fixed function devices
  - Beginning of product use --> Actel/Quicklogic FPGAs
  - Beginning of usage epoch --> (Re)configurable FPGAs
  - Every cycle --> traditional Instruction Set Processors

Granularity of Computational Elements

The FPGA Approach: The Logic Level

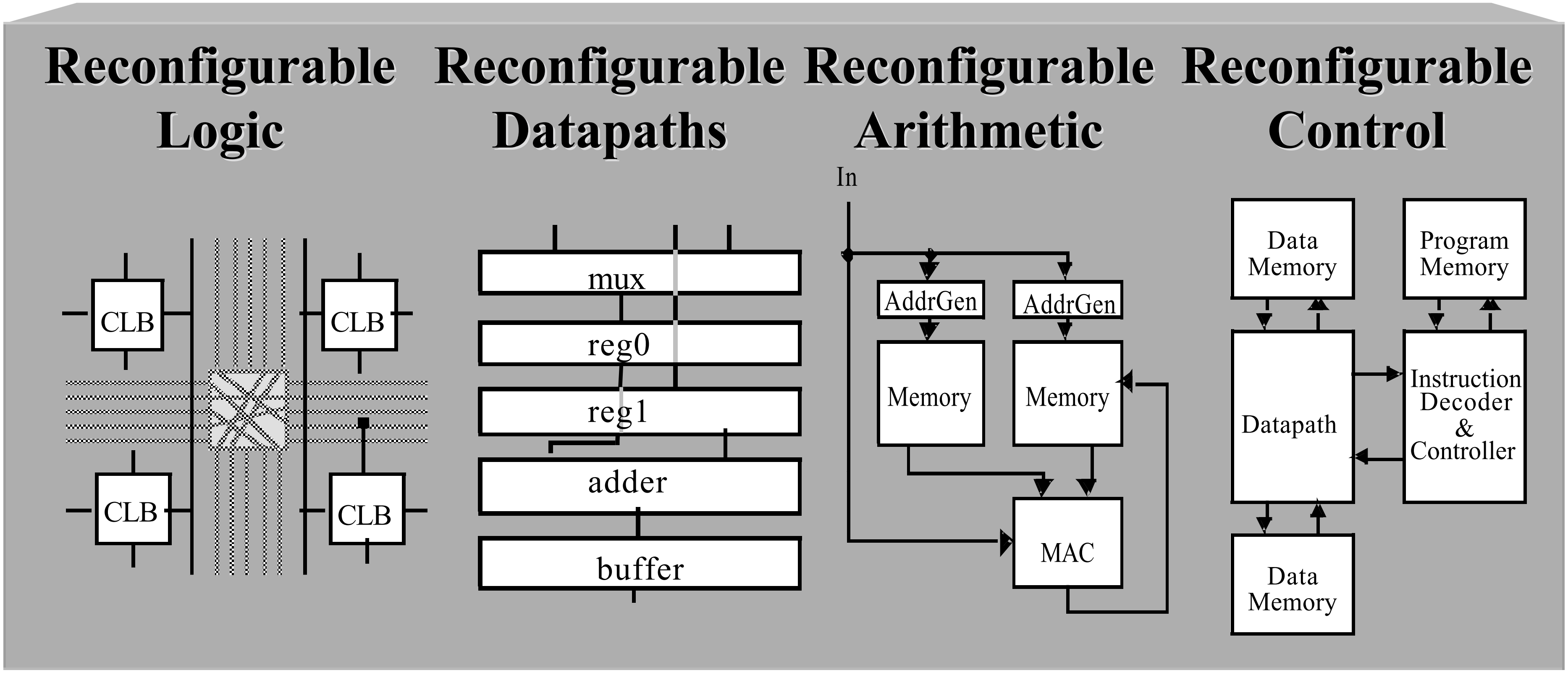


In	Out
00	0
01	1
10	1
11	0



2-LUT

Granularity of Computational Elements



Bit-Level Operations e.g. encoding	Dedicated data paths e.g. Filters, AGU	Arithmetic kernels e.g. Convolution	RTOS Process management

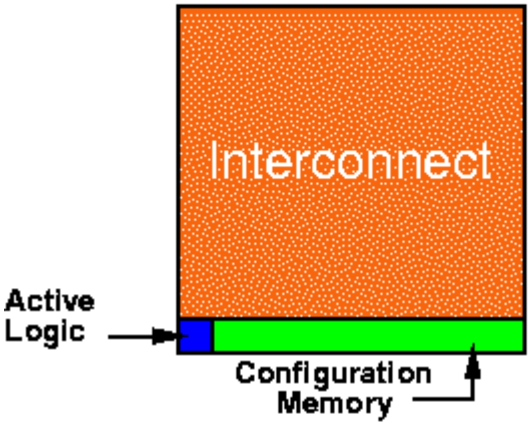
For Spatial Architectures

- **Interconnect dominant**
  - area
  - power
  - time

- **…so need to understand in order to optimize architectures**

Dominant in Area

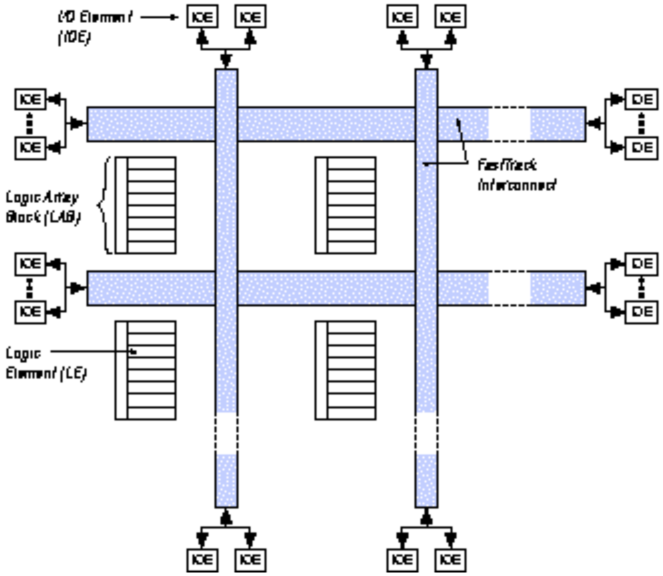
$$A_{bit\_elm} = A_{fixed} + \underbrace{N_{SW}(N_p, w, p) \cdot A_{SW}}_{\text{interconnect}} + \underbrace{\left(\frac{c}{w}\right) \cdot n_{ibits} \cdot A_{mem\_cell}}_{\text{instruction memory}} + \underbrace{d \cdot A_{mem\_cell}}_{\text{retiming memory}}$$



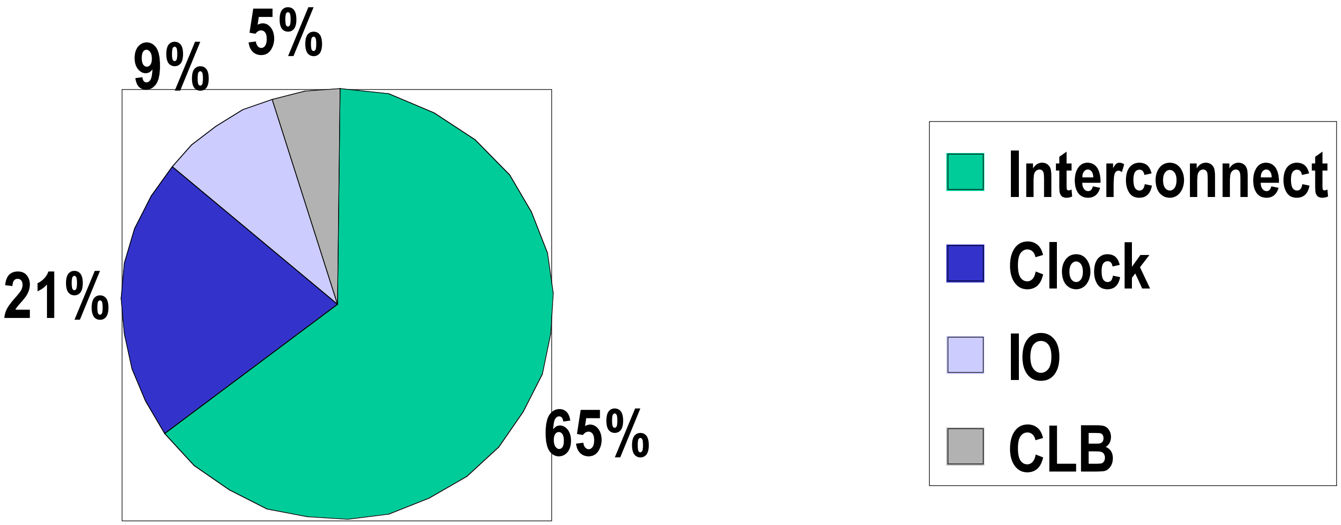
Function	Area (λ^2)	
LUT MUX + ff	20K	(generous, closer to 10K)
Programming Memory	80K	(240K typical unencoded)
Interconnect	700K	(for $N_p = 2048$)

Dominant in Time

Design	Path	Total Delay	LUT Delay	Inter. %
Altera 10K130V-2	LUT-local-LUT	2.5 ns	2.1 ns	16%
	LUT-row-local-LUT	6.6 ns	2.1 ns	68%
	LUT-column-local-LUT	11.1 ns	2.1 ns	81%
	LUT-row-column-local-LUT	15.6 ns	2.1 ns	87%
	LUT-row-fanout-local-LUT (fanout)	28 ns	2.1 ns	90%



Dominant in Power



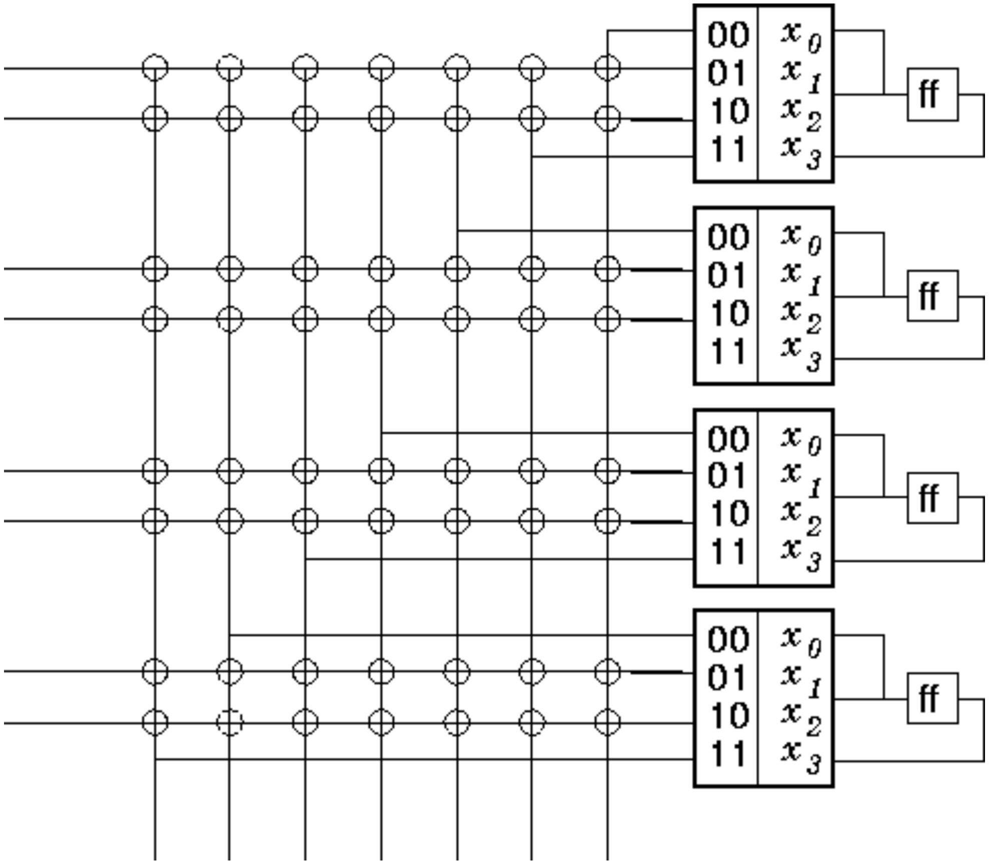
XC4003A data from Eric Kusse (UCB MS 1997)

Interconnect Design Issues

- Flexibility **-- route "anything"**
  - (within reason?)
- Area **-- wires, switches**
- Delay **-- switches in path, stubs, wire length**
- Power **-- switch, wire capacitance**
- Routability **-- computational difficulty finding routes**

A Naïve Approach: Crossbar

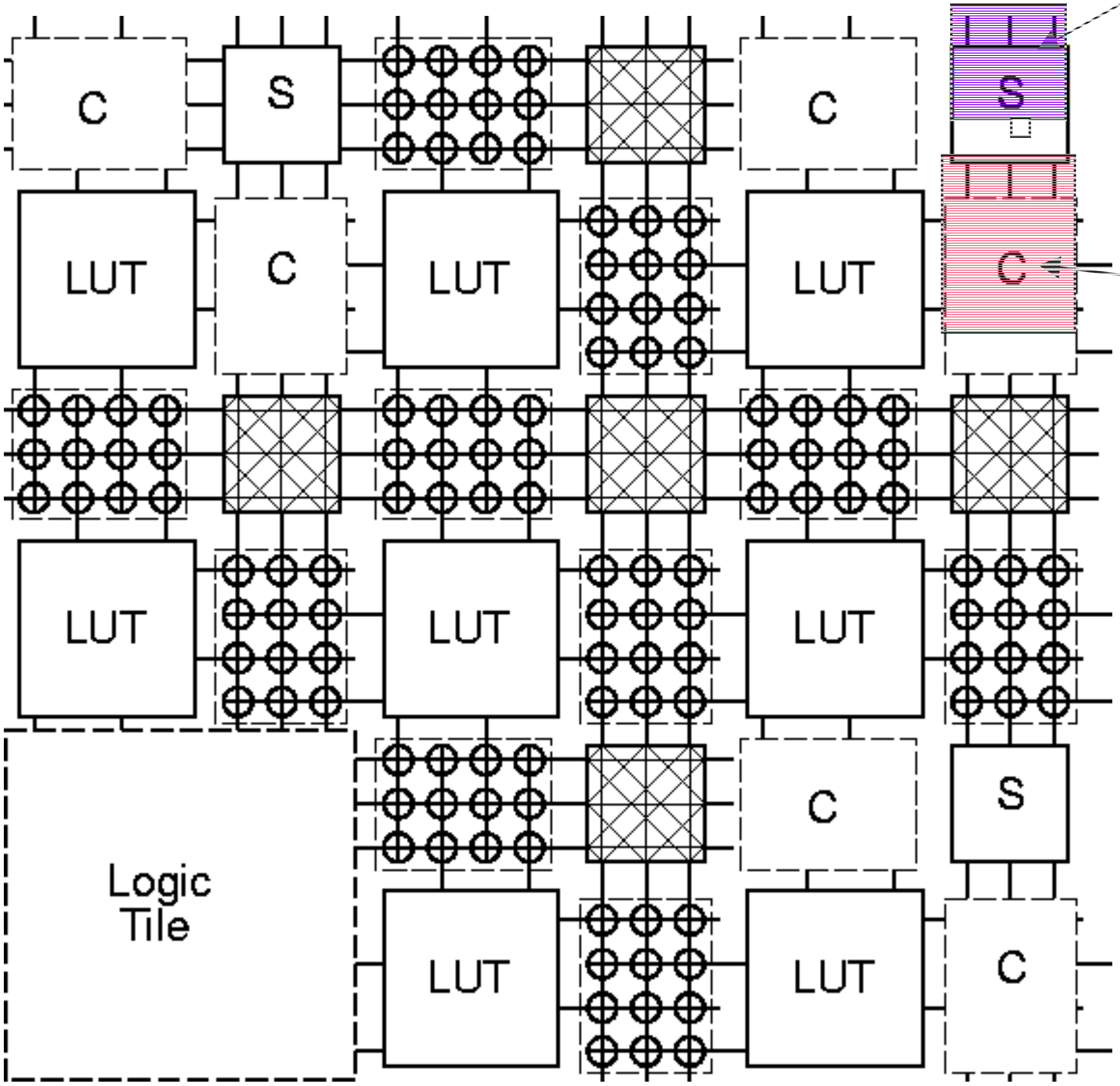
- **Any operator may consume output from any other operator**



Avoiding Crossbar Costs

- **Good architectural design**
  - Optimize for the common case
- **Designs have spatial locality**
- **We have freedom in operator placement**
- **Thus: Place connected components “close” together**
  - don’t need full interconnect?

Exploiting Locality – The Mesh



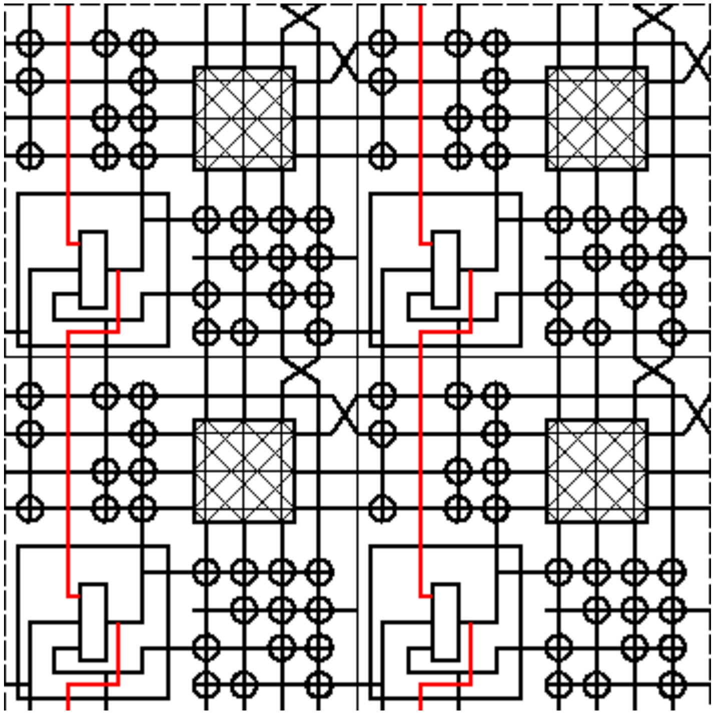
Switch Box

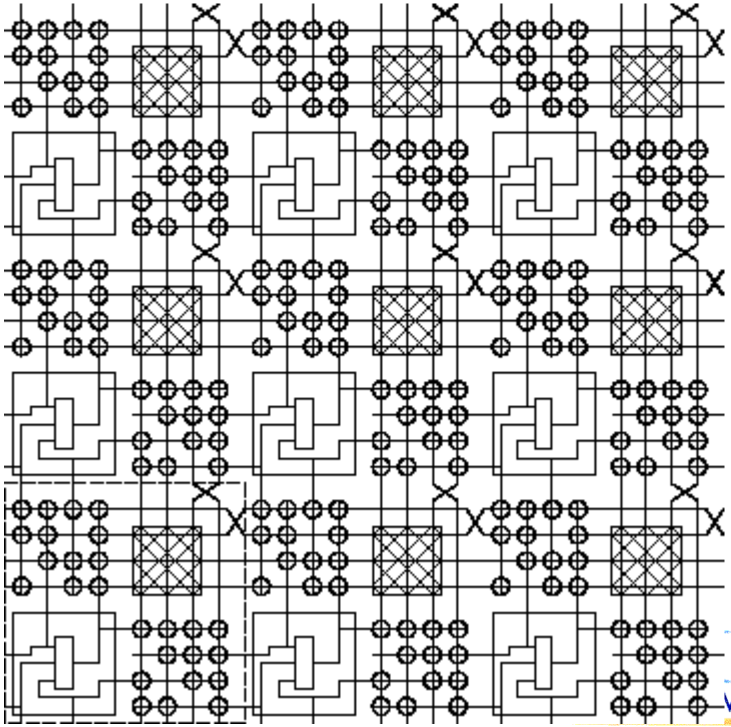
Connect Box

Meshes don't scale

Typical Extensions

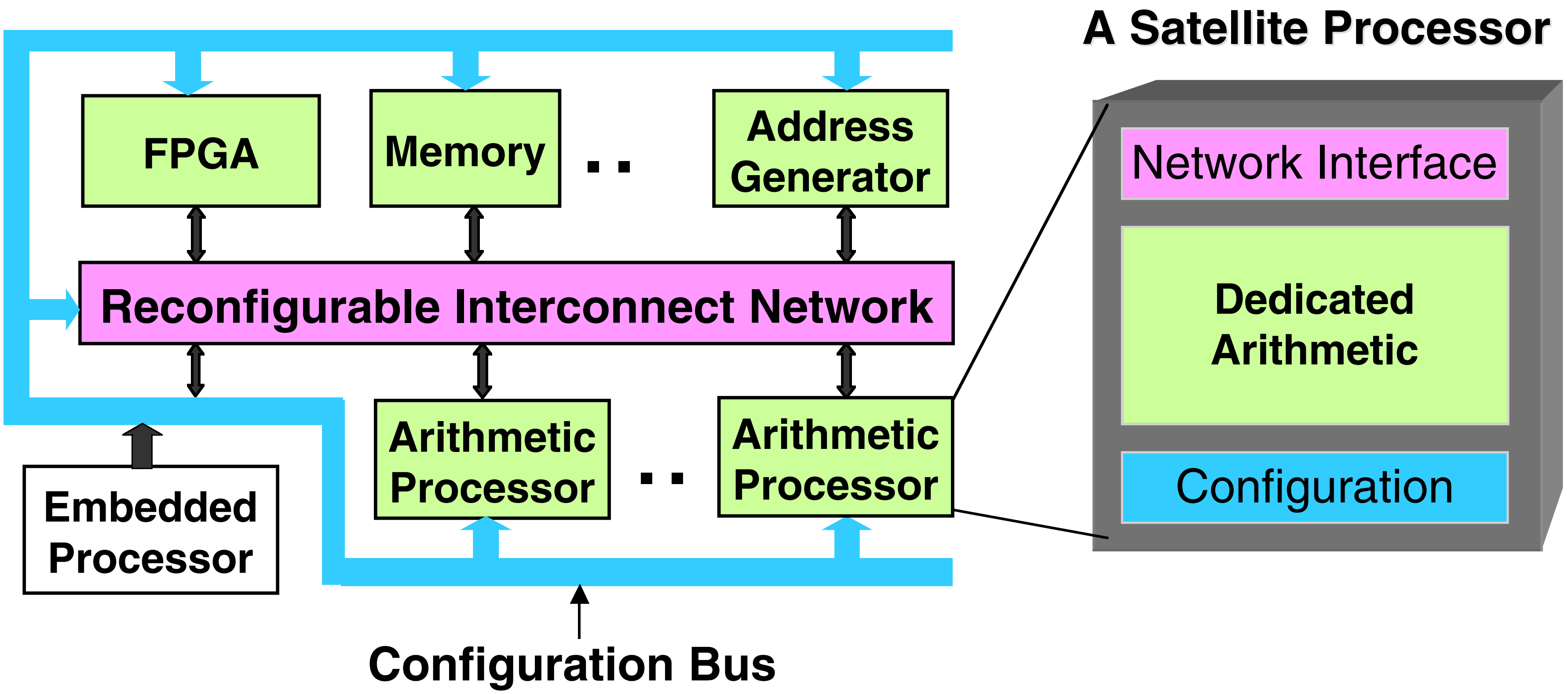
- Local neighbor-to-neighbor Interconnections
- Segmented Interconnect
- Hierarchical Network (tree, mesh)





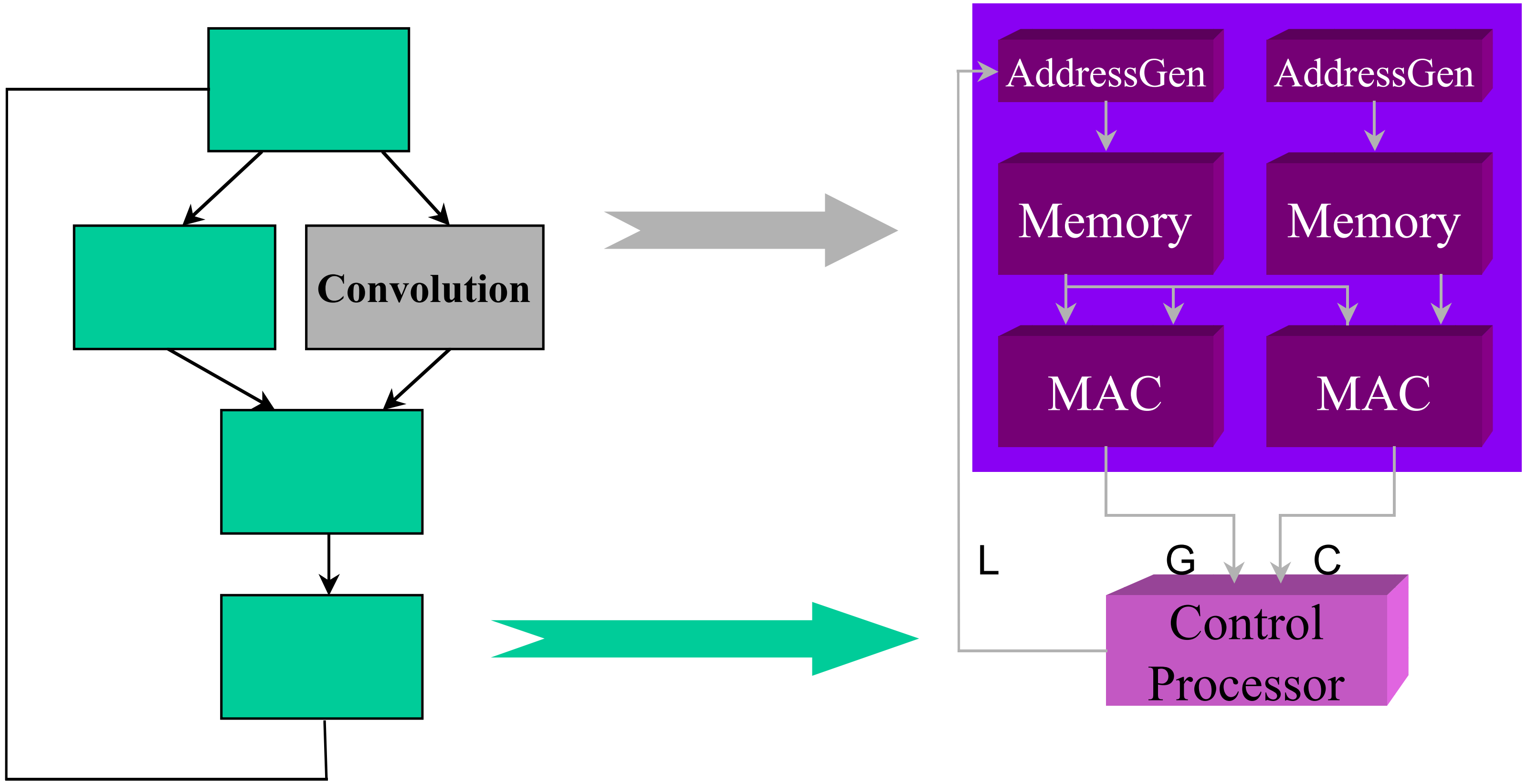
Wireless Research Center

Example 1:
The Pleiades Reconfigurable Architecture



- Computational kernels are “spawned” to satellite processors
- Control processor supports RTOS and reconfiguration
- Order(s) of magnitude energy-reduction over traditional programmable architectures

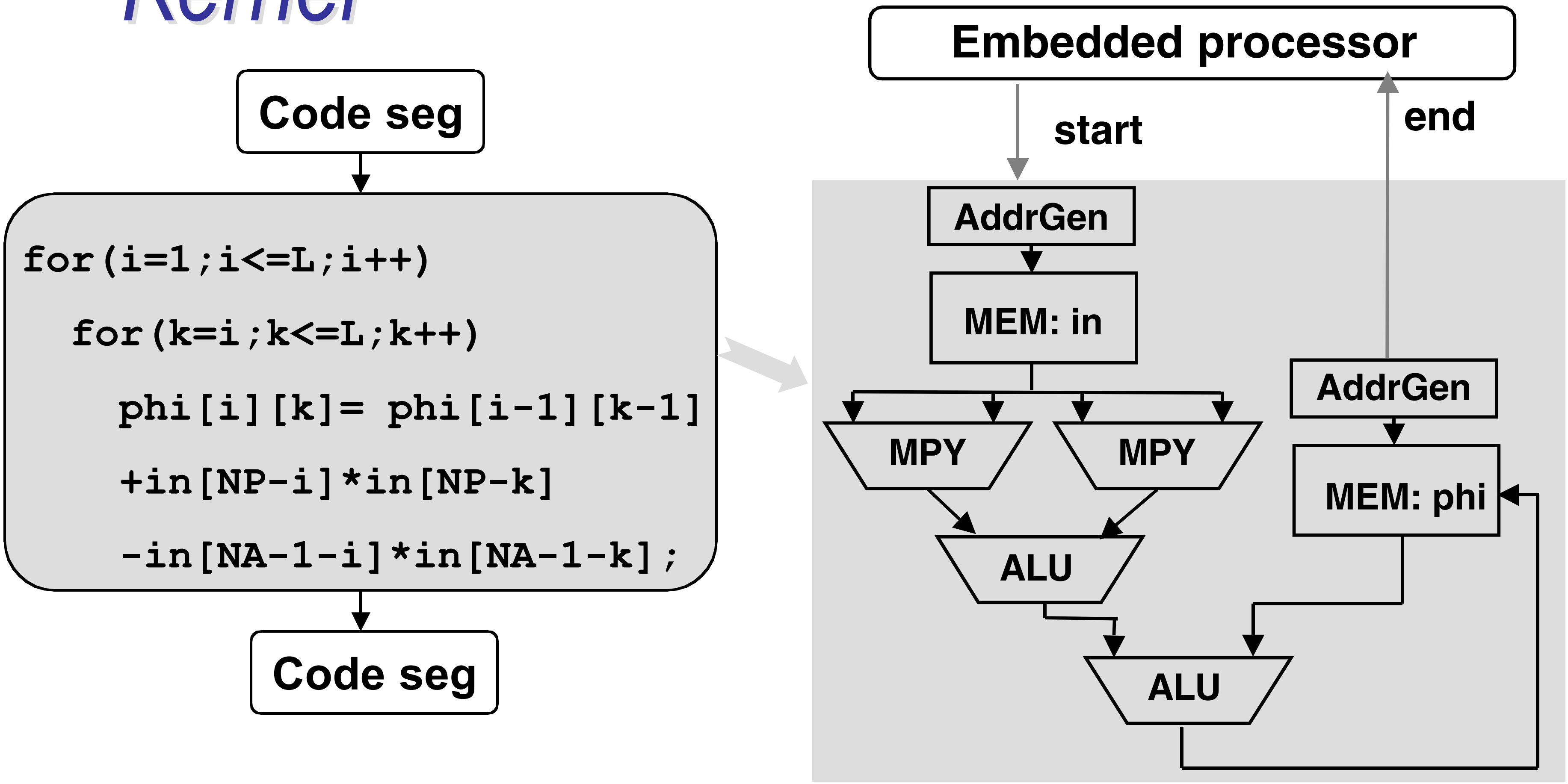
Matching Computation and Architecture



Two models of computation:
communicating processes + data-flow

Two architectural models:
sequential control+ data-driven

Execution Model of a Data-Flow Kernel



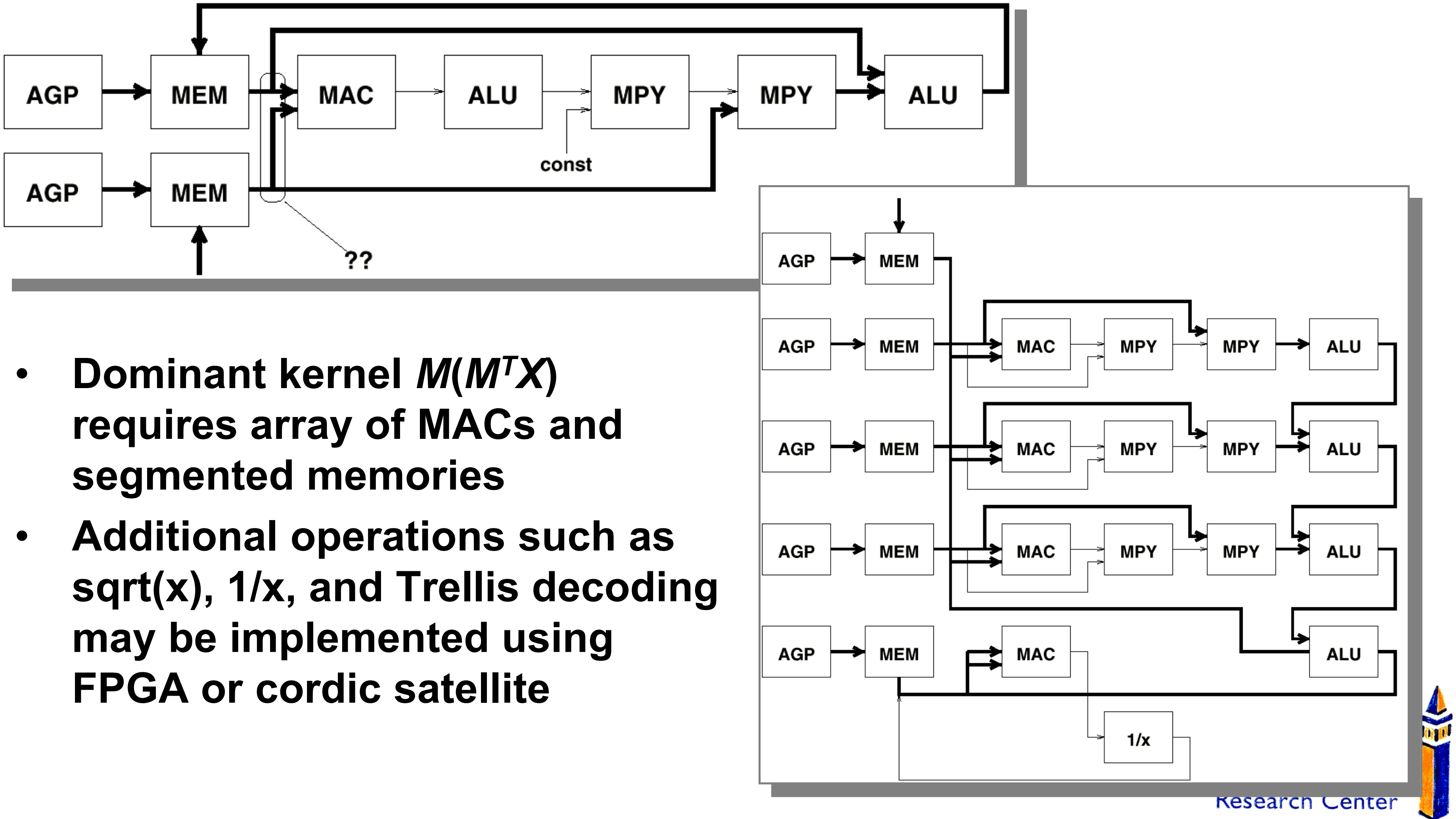
Code seg

```
for(i=1;i<=L;i++)
  for(k=i;k<=L;k++)
    phi[i][k]= phi[i-1][k-1]
    +in[NP-i]*in[NP-k]
    -in[NA-1-i]*in[NA-1-k];
```

Code seg

- **Distributed control and memory**

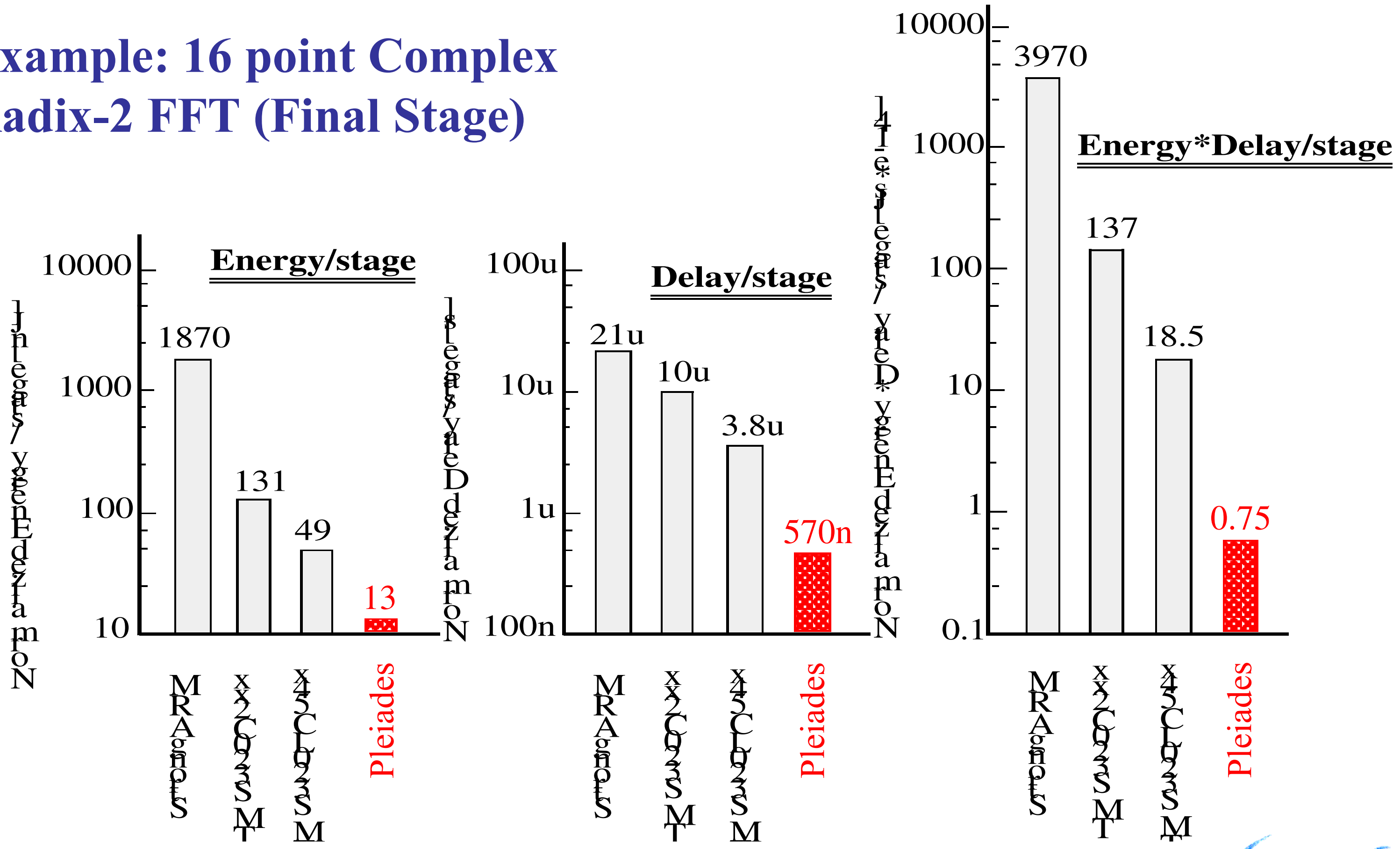
Reconfigurable Kernels for W-CDMA



- **Dominant kernel $M(M^TX)$ requires array of MACs and segmented memories**
- **Additional operations such as sqrt(x), 1/x, and Trellis decoding may be implemented using FPGA or cordic satellite**

Impact of Architectural Choice

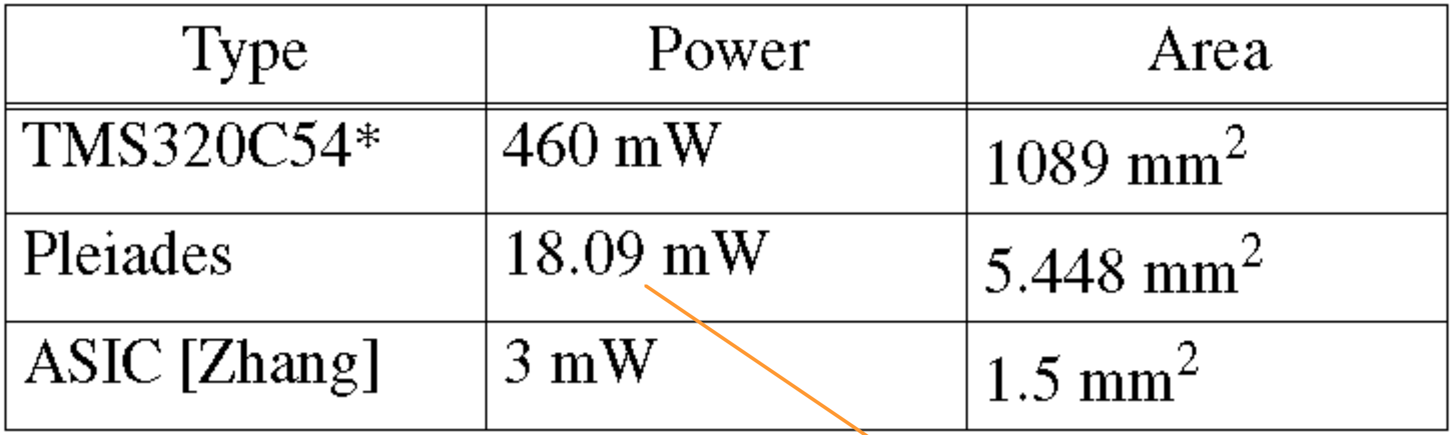
Example: 16 point Complex Radix-2 FFT (Final Stage)



Berkeley Wireless Research Center

Architecture Comparison

LMS Correlator at 1.67 MSymbols Data Rate
Complexity: 300 Mmult/sec and 357 Macc/sec



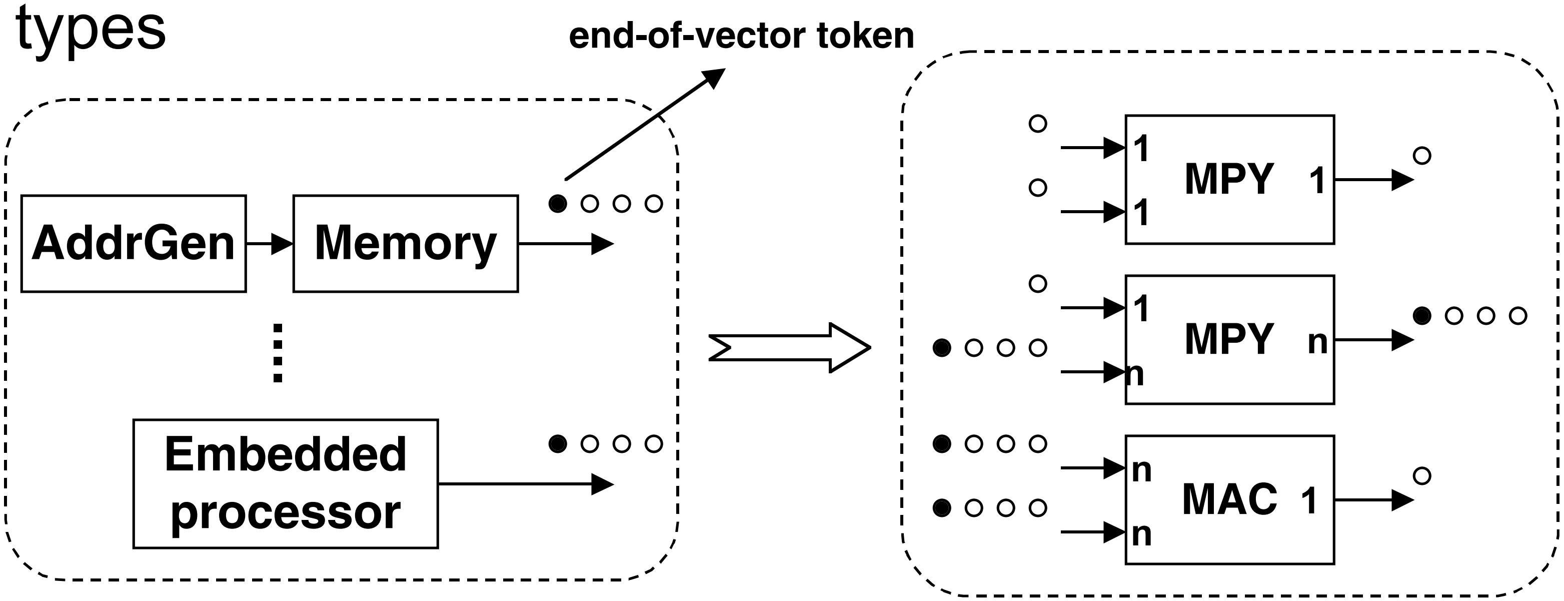
Type	Power	Area
TMS320C54*	460 mW	1089 mm^2
Pleiades	18.09 mW	5.448 mm^2
ASIC [Zhang]	3 mW	1.5 mm^2

16 Mmacs/mW!

Note: TMS implementation requires 36 parallel processors to meet data rate - validity questionable

Inter-Satellite Communication

- Data-driven execution
  - A satellite processor is enabled only when input data is ready
- Data sources generate data of different types: scalars, vectors, matrices
- Data computing processors handle data inputs of different types



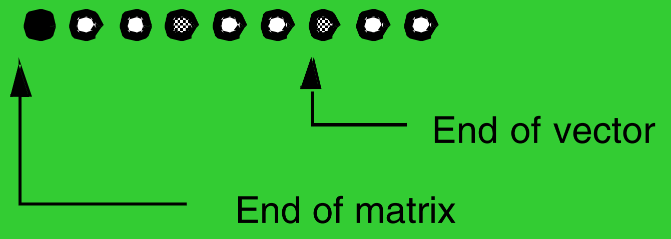
Data sources

Data computing processors

AGP Satellite

- Address generator for the SRAM satellite
- Generates data streams of different types; all other satellites process data streams
- Uses loop counters and stride counters to support 2 levels of nesting
- Control information sent in parallel with the data using 2 additional control bits

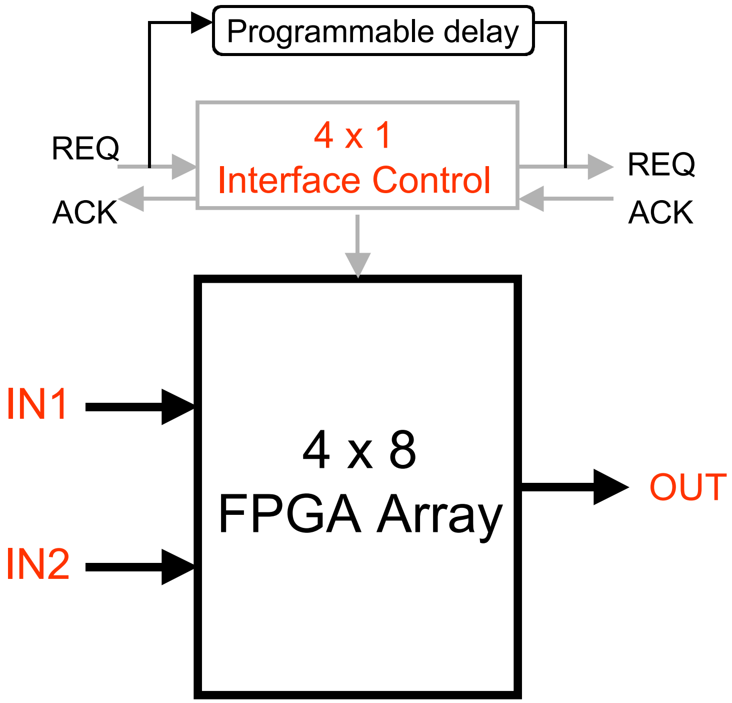
```
for n=1 to 3 {
   for k=1 to 3 {
      addr_read(n,k);
   }
}
```



End of vector

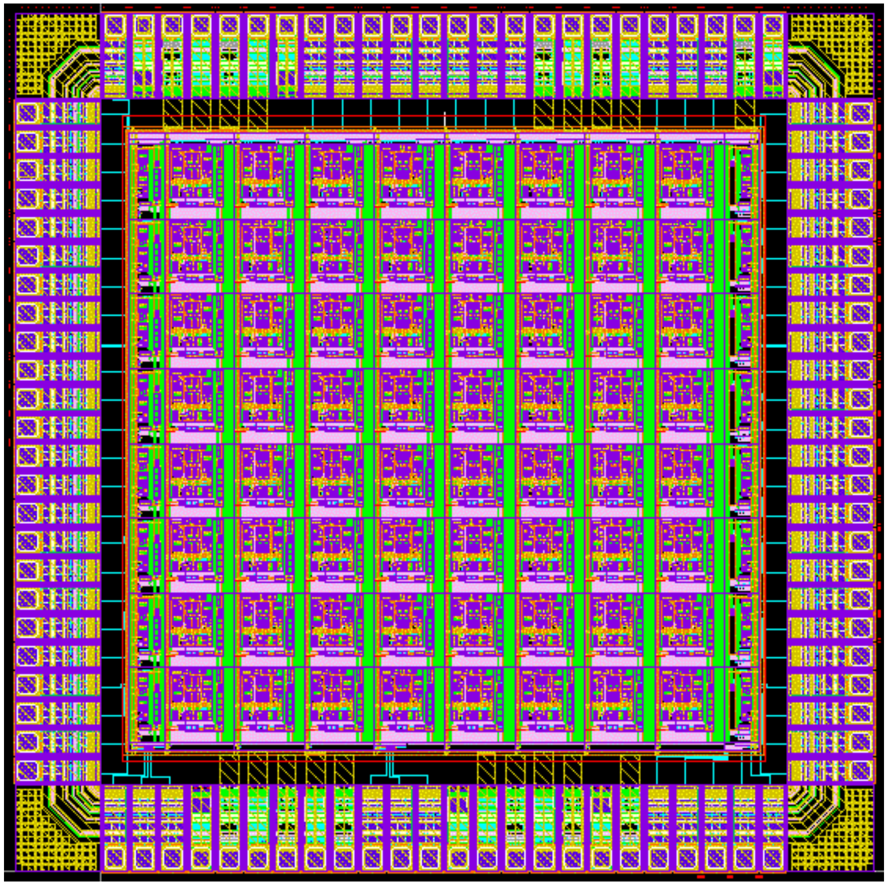
End of matrix

Satellite Processors: FPGA



- Reconfigurable for both logic function and interface control
- 4 x 9 CLB array in total
- 5-input 3-output CLBs
- 3 levels of interconnect hierarchy
- Mapped to various arithmetic functions and control
- Programmable clock generator

Low-Energy Embedded FPGA

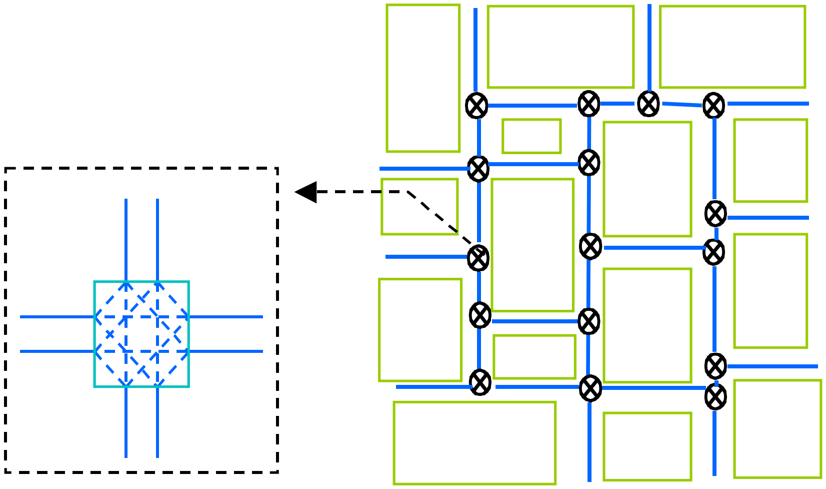


- **Test chip**
  - 8x8 CLB array
  - 5 in - 3 out CLB
  - 3-level interconnect hierarchy
  - 4 mm^2 in 0.25 μm ST CMOS
  - 0.8 and 1.5 V supply
- **Simulation Results**
  - 125 MHz Toggle Frequency
  - 50 MHz 8-bit adder
  - energy 70 times lower than comparable Xilinx
- **Parameterized module generator available**

Berkeley Wireless Research Center

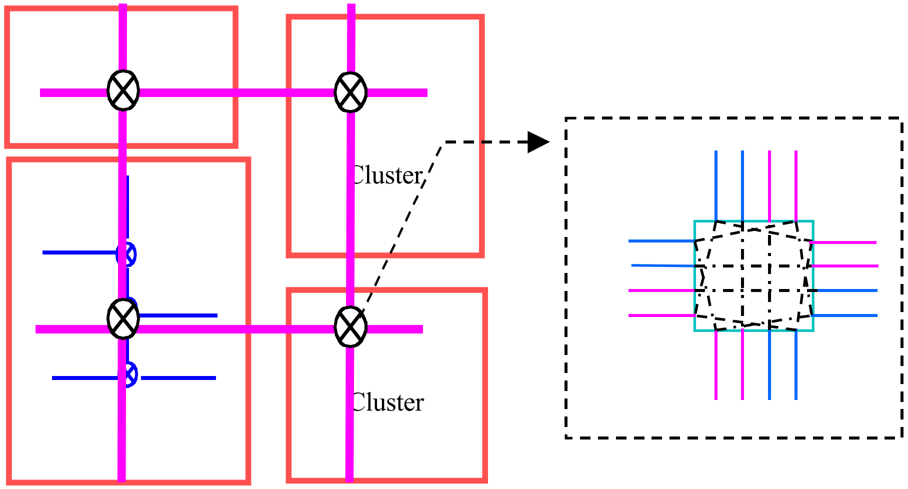
Reconfigurable Interconnect Network

Level-1 Mesh



Universal Switchbox

Level-2 Mesh



Hierarchical Switchbox

Irregular mesh for Heterogeneous blocks

- A channel along every side of each block
- A switch box at every cross-point

Building hierarchy by clustering

- *Intra-cluster*: mesh structure
- *Inter-cluster*: larger-granularity mesh

Saves energy by a factor of 7 compared to straightforward crossbar network!

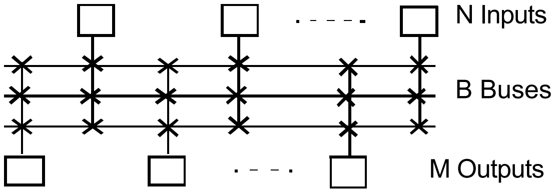
Fast Design Space Exploration
Interconnect Models

		dot_ product	vector sum w/ scalar mult.	IIR
Multi-bus		50	50	138
Mesh	*Best*	8.7	5.2	21.6
	Worst	17.7	14.7	43.4
H. Mesh	***Best***	**4.7**	**3.8**	**18.8**
	Worst	**11.1**	**10.2**	**31.3**

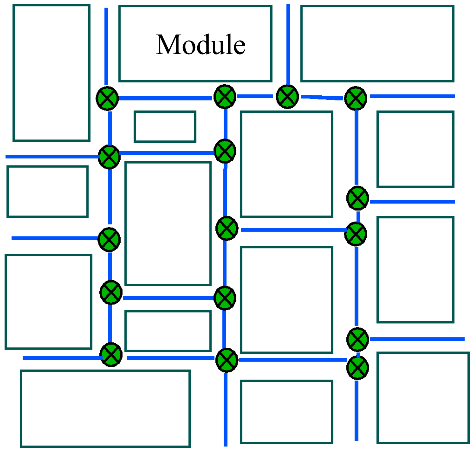
Model:

- **Interconnect energy and delay model**
- **Algorithm mapping**
- **Graph-based place and route**

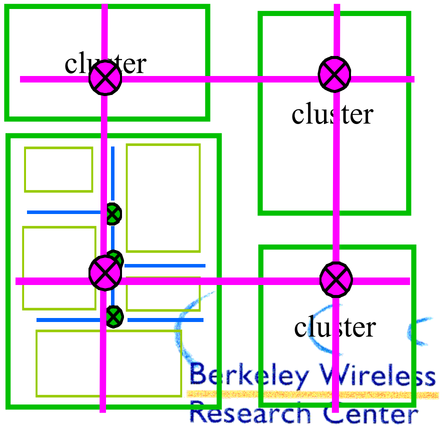
Multi-Bus



Mesh

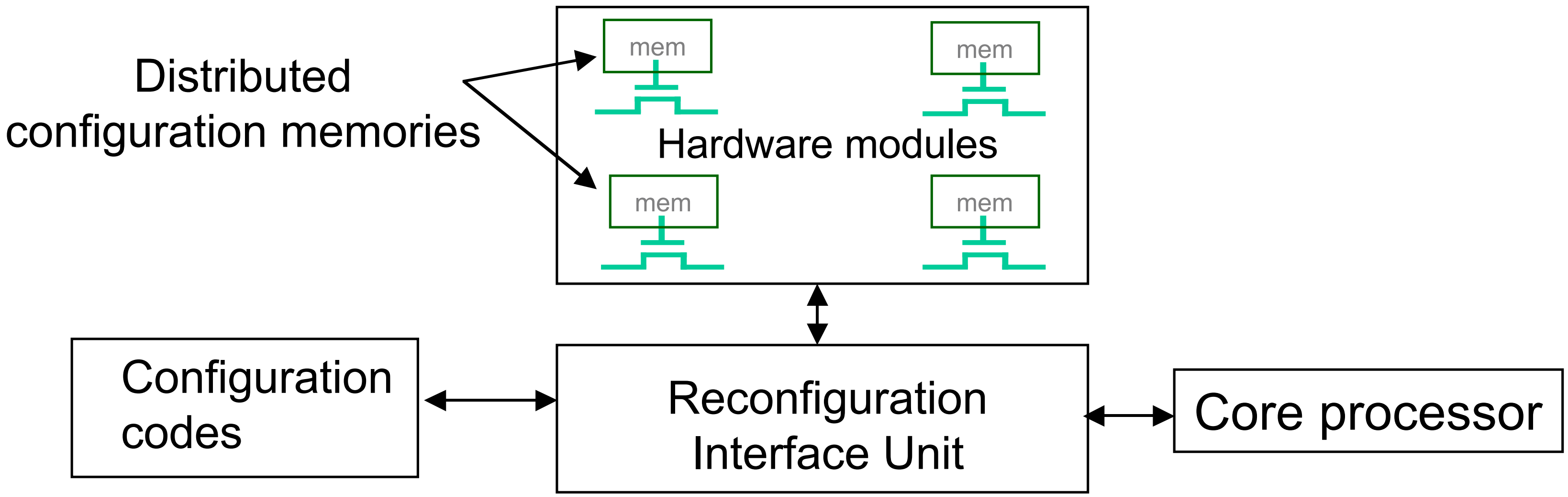


Hierarchical Mesh



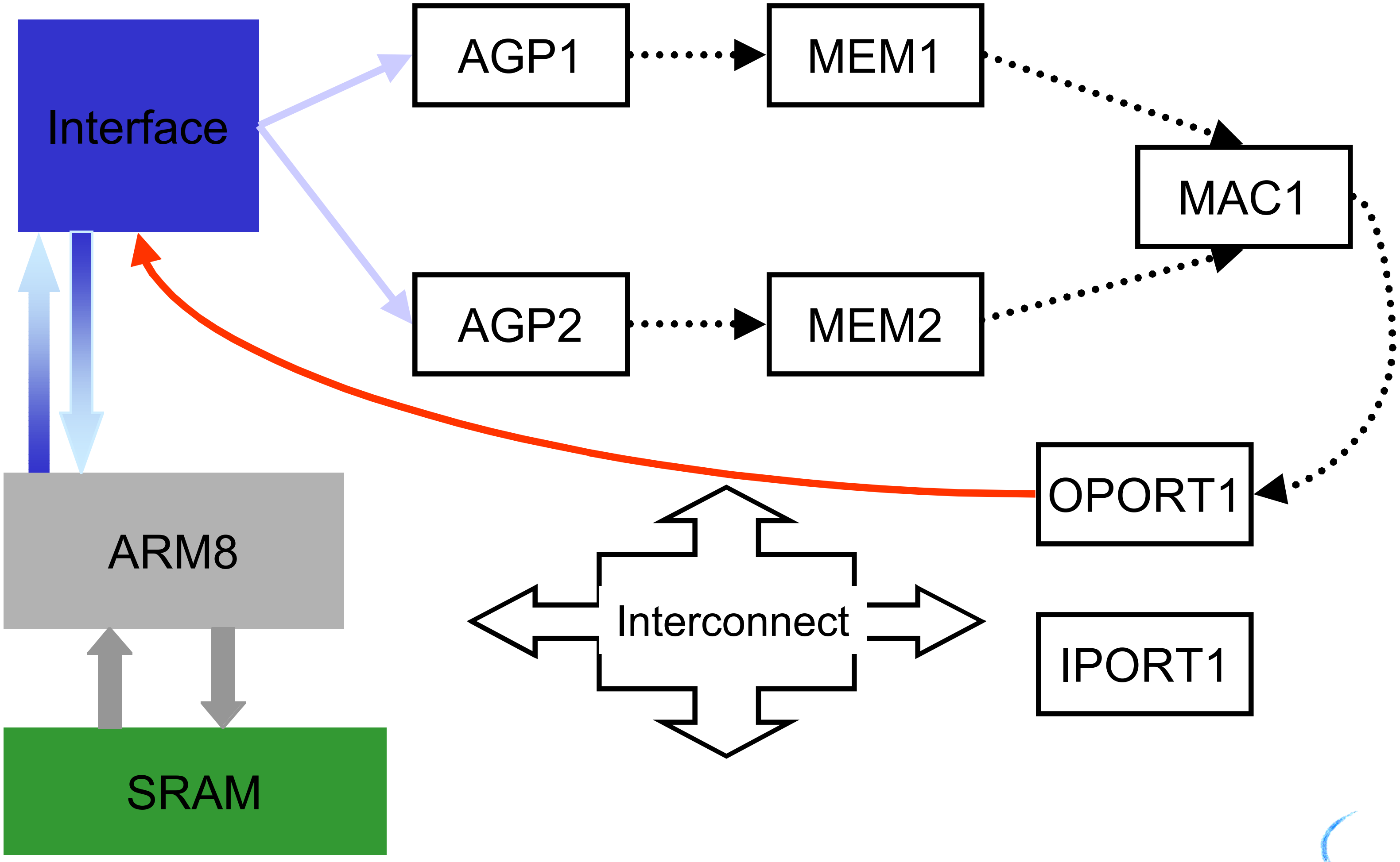
Berkeley Wireless Research Center

Reconfiguration Model

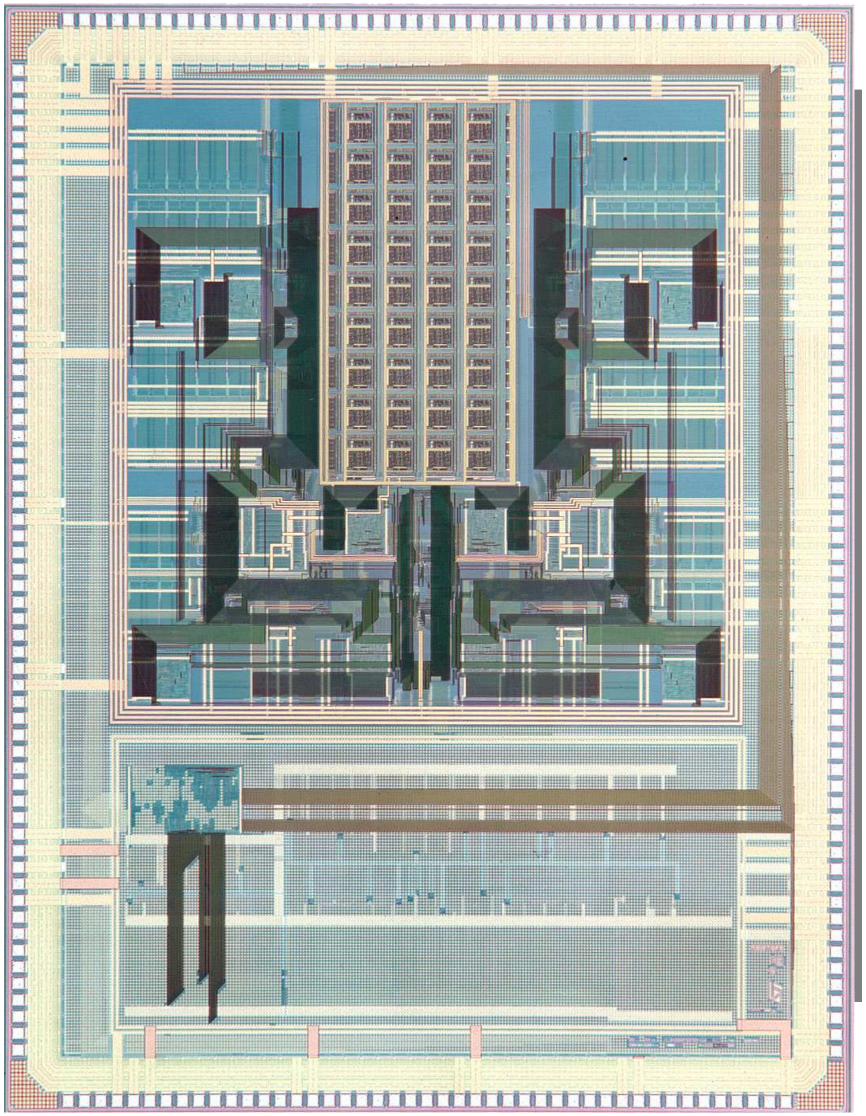

Distributed configuration memories
mem
mem
Hardware modules
mem
mem
Configuration codes
Reconfiguration Interface Unit
Core processor

- **Configuration codes are created statically at compile time**
- **Every configuration memory is reset and rewritten with new configuration code before each kernel**
- **The core processor uses memory read/write instructions to perform the reconfiguration**

Kernel Execution & Configuration



Maia: Reconfigurable Baseband Processor for Wireless

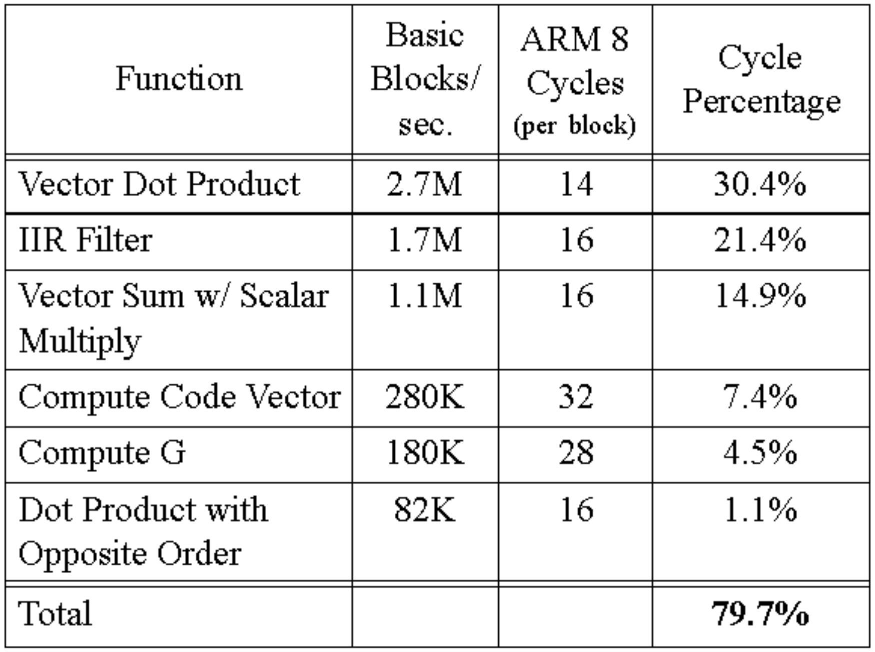


- 0.25um tech: 4.5mm x 6mm
- 1.2 Million transistors
- 40 MHz at 1V
- 1 mW VCELP voice coder
- Hardware
  - 1 ARM-8
  - 8 SRAMs & 8 AGPs
  - 2 MACs
  - 2 ALUs
  - 2 In-Ports and 2 Out-Ports
  - 14x8 FPGA

Berkeley Wireless Research Center

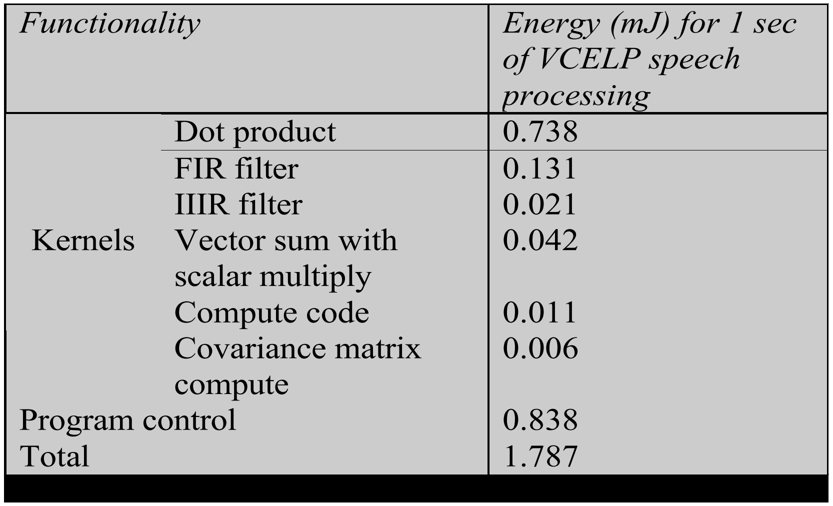
Results of VCELP Voice Coder

VCELP code breakdown



Function	Basic Blocks/ sec.	ARM 8 Cycles (per block)	Cycle Percentage
Vector Dot Product	2.7M	14	30.4%
IIR Filter	1.7M	16	21.4%
Vector Sum w/ Scalar Multiply	1.1M	16	14.9%
Compute Code Vector	280K	32	7.4%
Compute G	180K	28	4.5%
Dot Product with Opposite Order	82K	16	1.1%
Total			**79.7%**

VCELP Energy breakdown



Functionality		*Energy (mJ) for 1 sec of VCELP speech processing*
Kernels	Dot product	0.738
	FIR filter	0.131
	IIIR filter	0.021
	Vector sum with scalar multiply	0.042
	Compute code	0.011
	Covariance matrix compute	0.006
Program control		0.838
Total		1.787

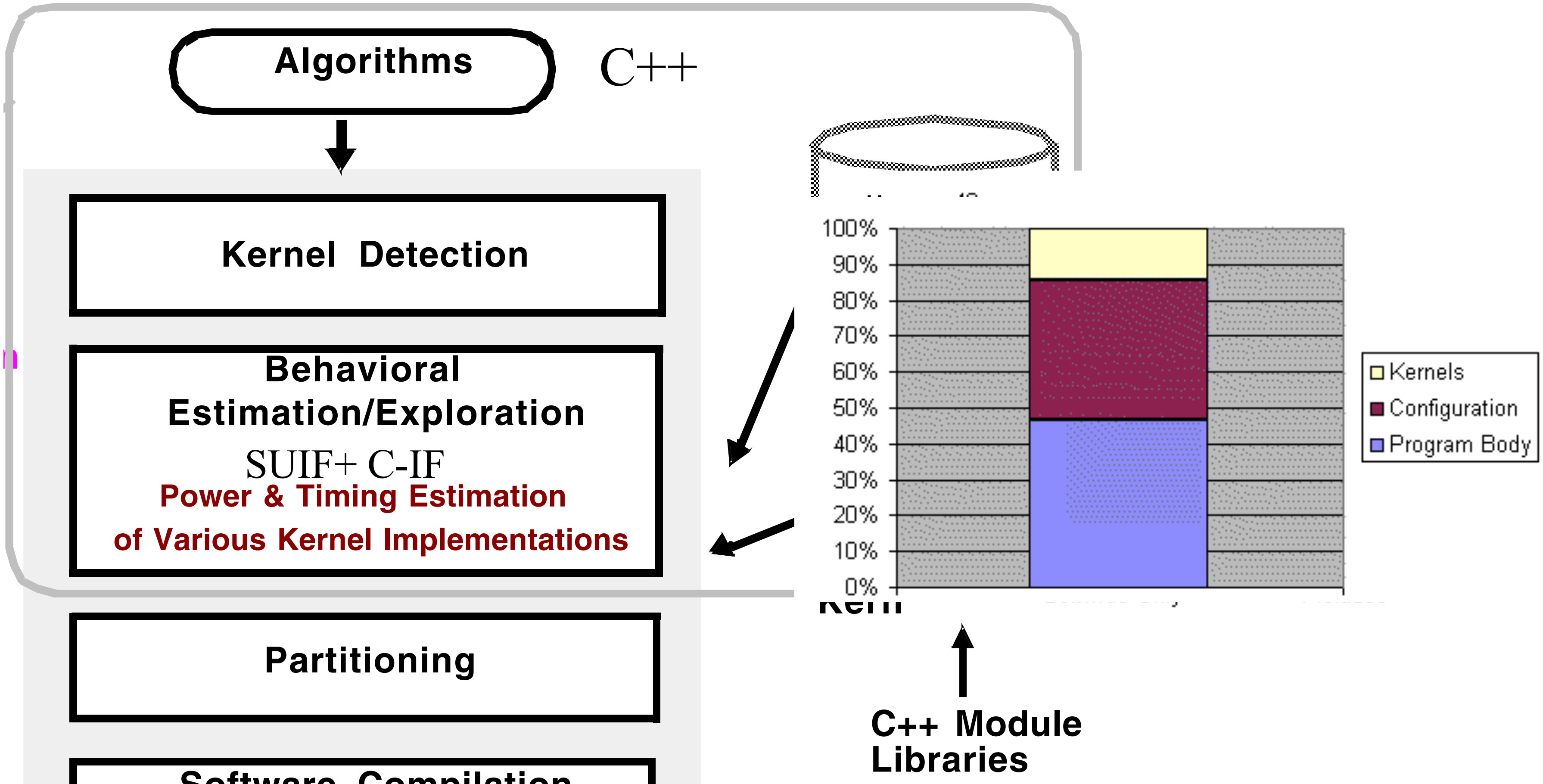
79.7% of VCELP Code maps onto Reconfigurable Datapath

Compared to state-of-art 17mW DSP

Design Methodology and Flow

- Requires architecture exploration over heterogeneous implementation fabrics
- Should support refinement and co-design of hardware and software, as well as behavior and architecture
- Should consider all important metrics, and present PDA (Power-Delay-Area) perspective

Software Methodology Flow



Algorithms

C++

Kernel Detection

Behavioral Estimation/Exploration

SUIF+ C-IF

Power & Timing Estimation of Various Kernel Implementations

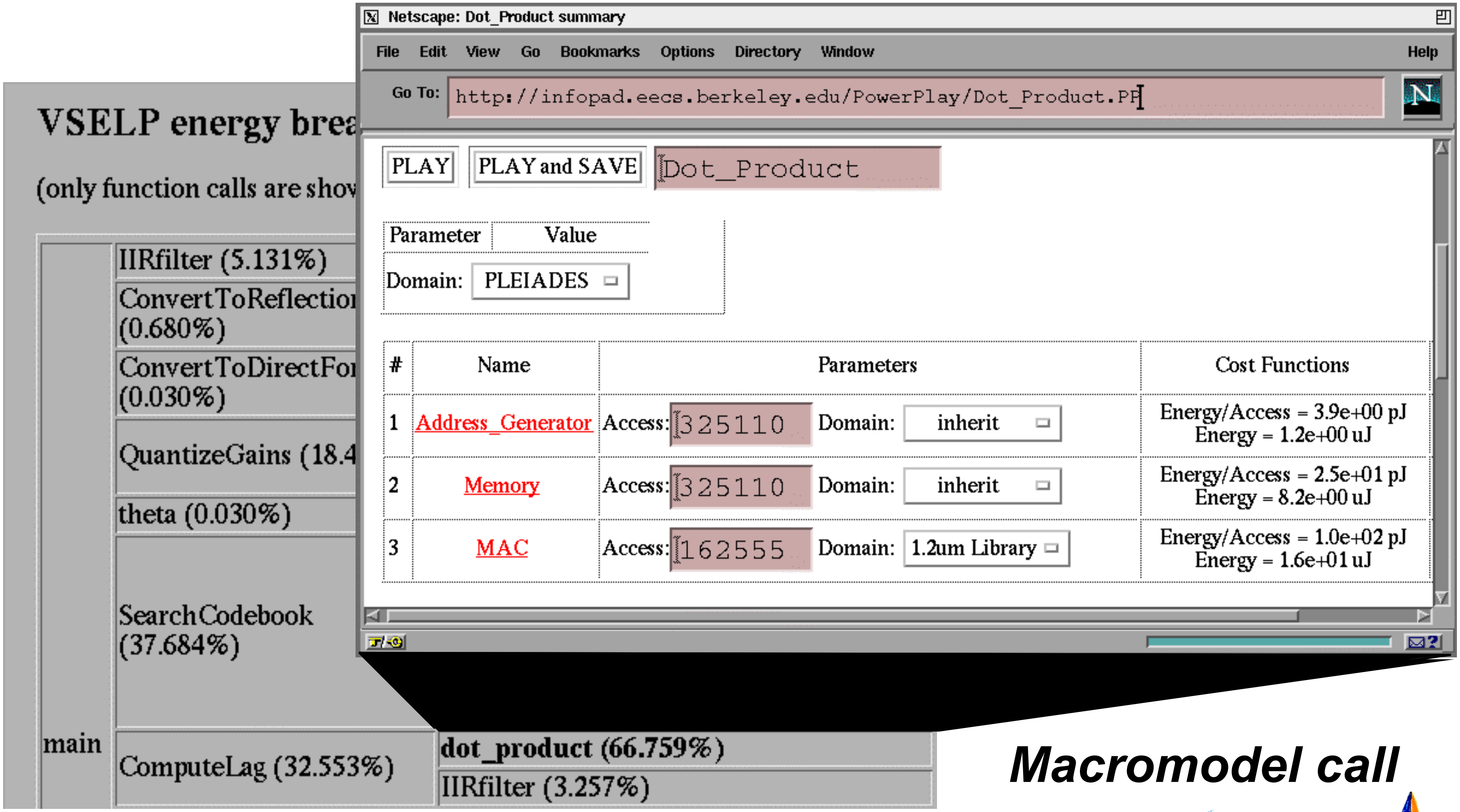
Partitioning

Software Compilation
Reconfig. Hardware Mapping
Interface Code Generation

C++ Module Libraries

Berkeley Wireless Research Center

Hardware-Software Exploration



VSELP energy brea

(only function calls are show

main

IIRfilter (5.131%)

ConvertToReflection (0.680%)

ConvertToDirectFor (0.030%)

QuantizeGains (18.4

theta (0.030%)

SearchCodebook (37.684%)

ComputeLag (32.553%)

dot_product (66.759%)

IIRfilter (3.257%)

Netscape: Dot_Product summary
File Edit View Go Bookmarks Options Directory Window Help
PLAY
PLAY and SAVE

Go To: http://infopad.eecs.berkeley.edu/PowerPlay/Dot_Product.PP

Dot_Product

Parameter	Value
Domain:	PLEIADES

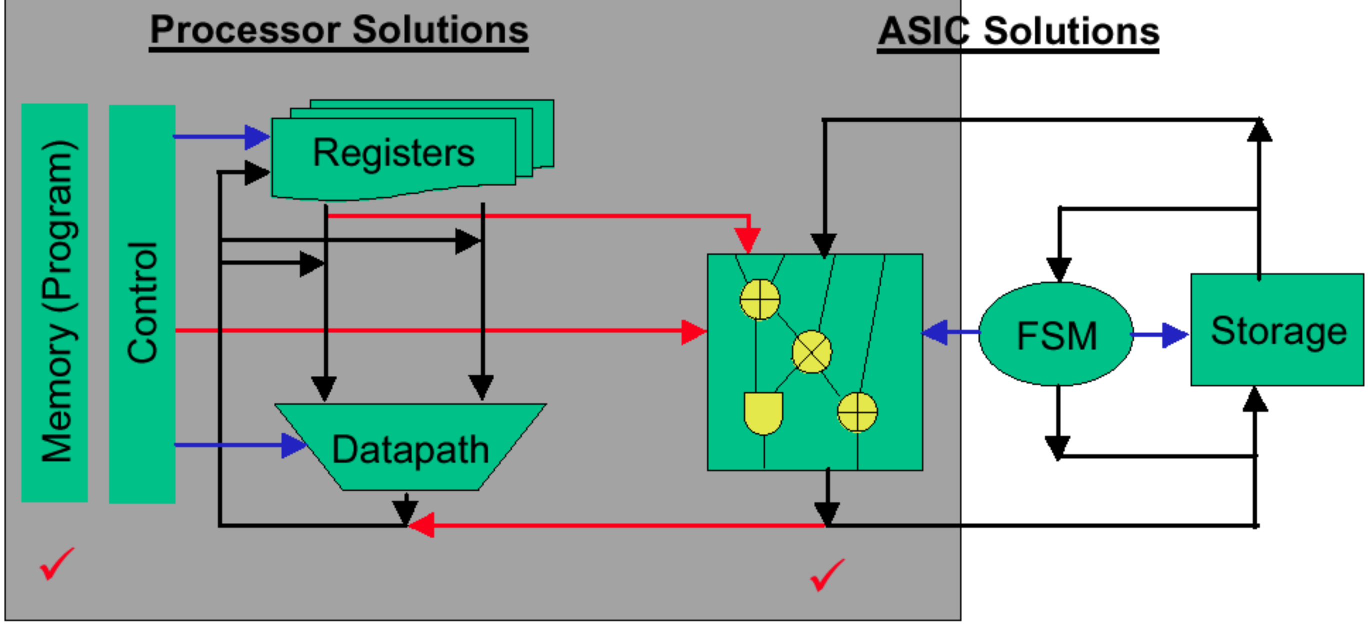
#	Name	Parameters				Cost Functions
1	Address_Generator	Access:	325110	Domain:	inherit	Energy/Access = 3.9e+00 pJ Energy = 1.2e+00 uJ
2	Memory	Access:	325110	Domain:	inherit	Energy/Access = 2.5e+01 pJ Energy = 8.2e+00 uJ
3	MAC	Access:	162555	Domain:	1.2um Library	Energy/Access = 1.0e+02 pJ Energy = 1.6e+01 uJ

Macromodel call

Industrial Example 1: Xtensa Configurable Processor

Core processor with extendible instruction set

Combines spatial and temporal processing

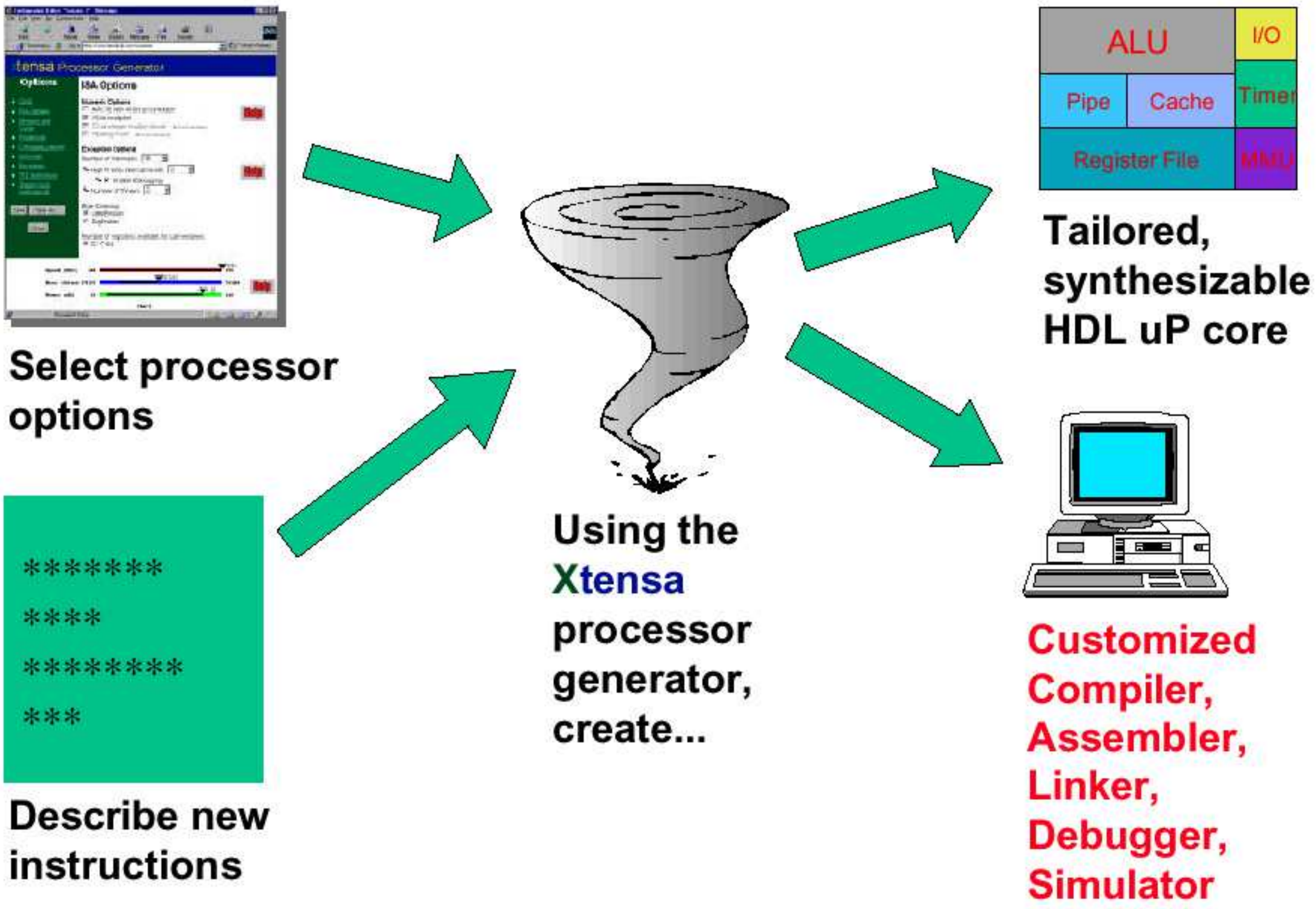


	Correct	Efficient
Processor	✓	
ASIC		✓

Small core: 0.7 mm2 in 0.18 mm
~ 3 MIPS/mW

Source: Tensilica, Inc

Design Methodology – a Crucial Component



Select processor options

Describe new instructions

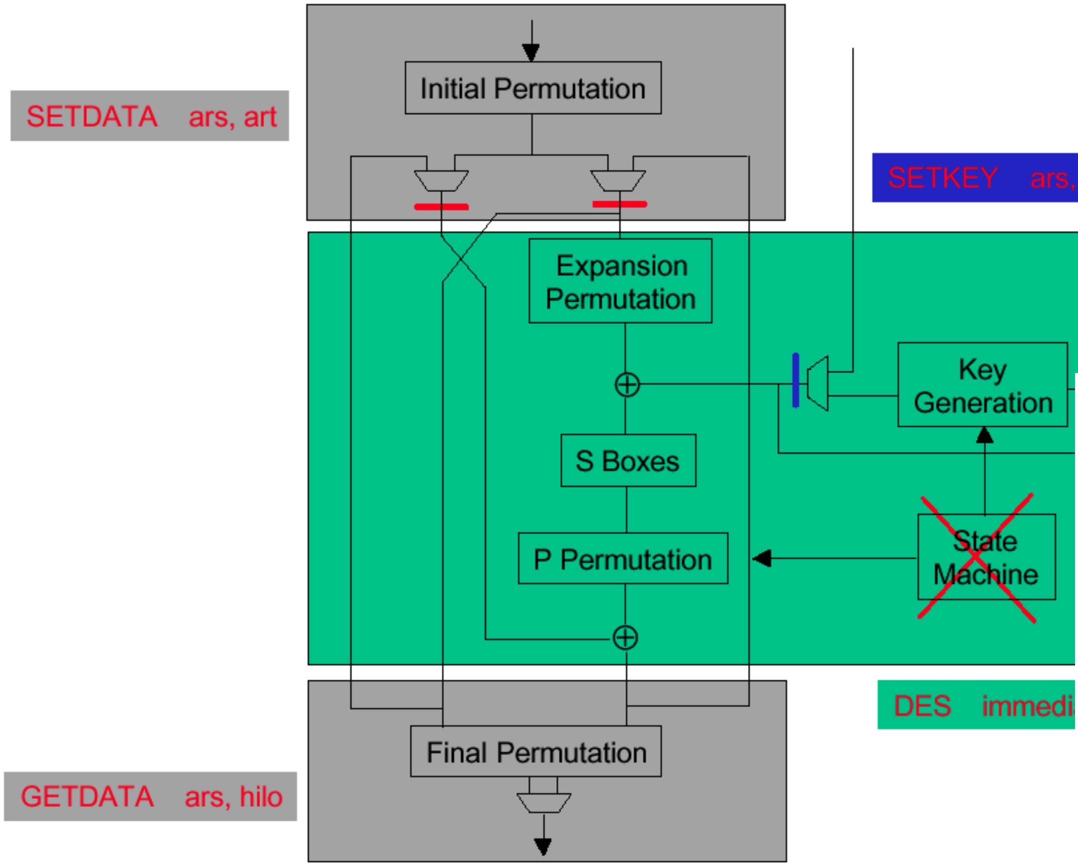
Using the **Xtensa** processor generator, create...

Tailored, synthesizable HDL uP core

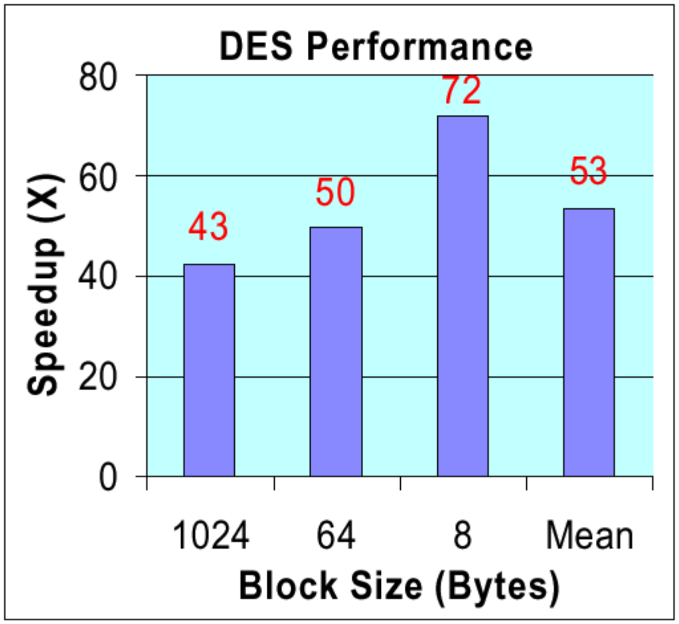
Customized Compiler, Assembler, Linker, Debugger, Simulator

Source: Tensilica, Inc

Example: A DES Encryption Extension

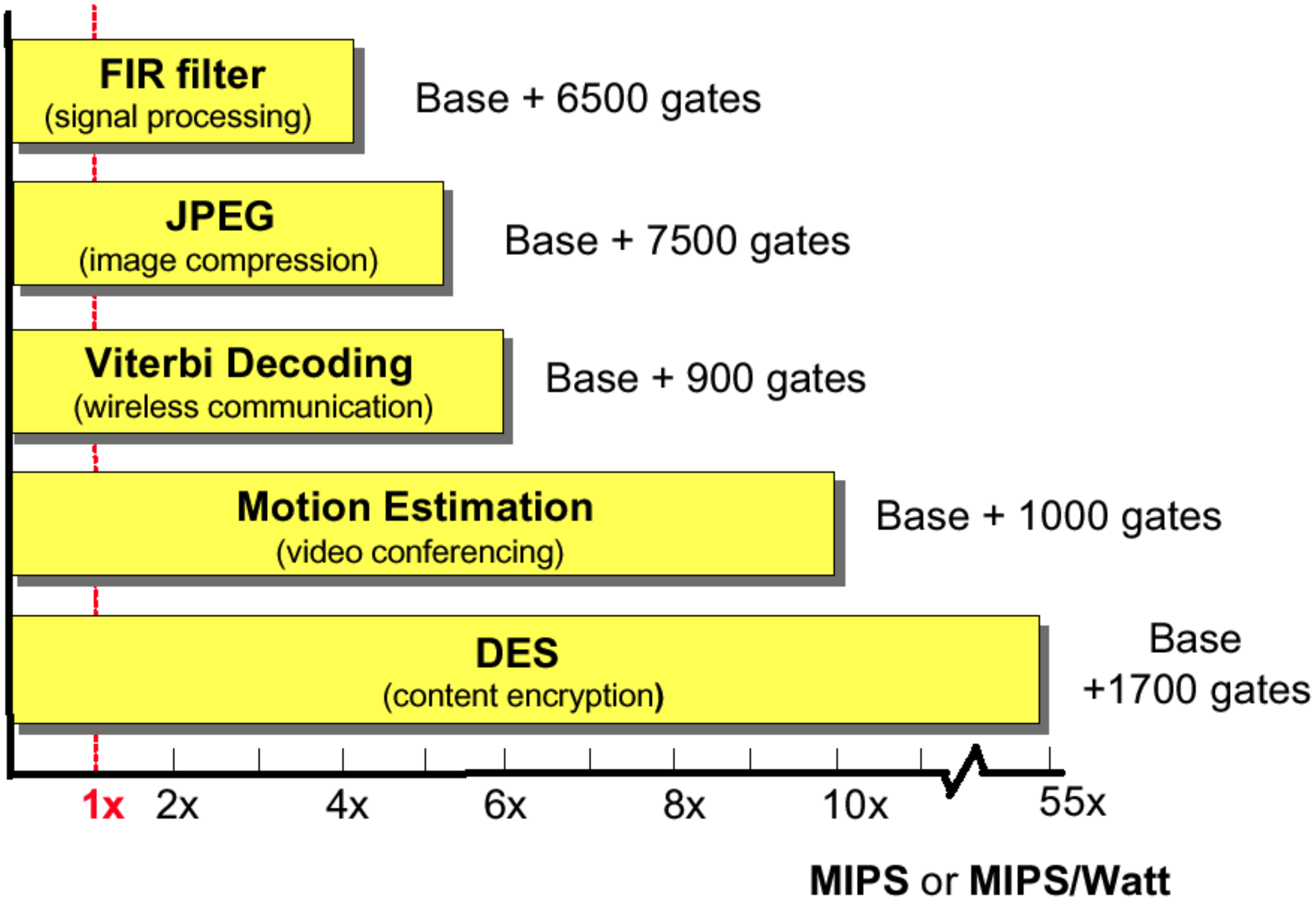


4 extra instructions
1700 additional gates
No cycle time impact
Code size reduction



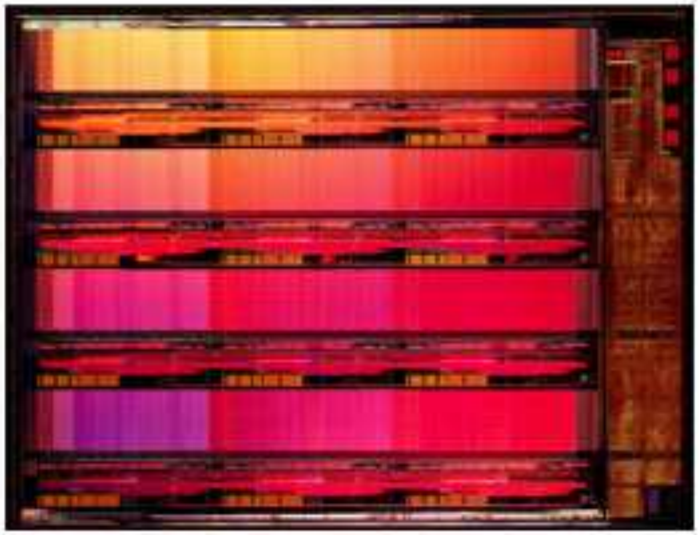
Source: Tensilica, Inc

Improvement over GP 32-bit processor

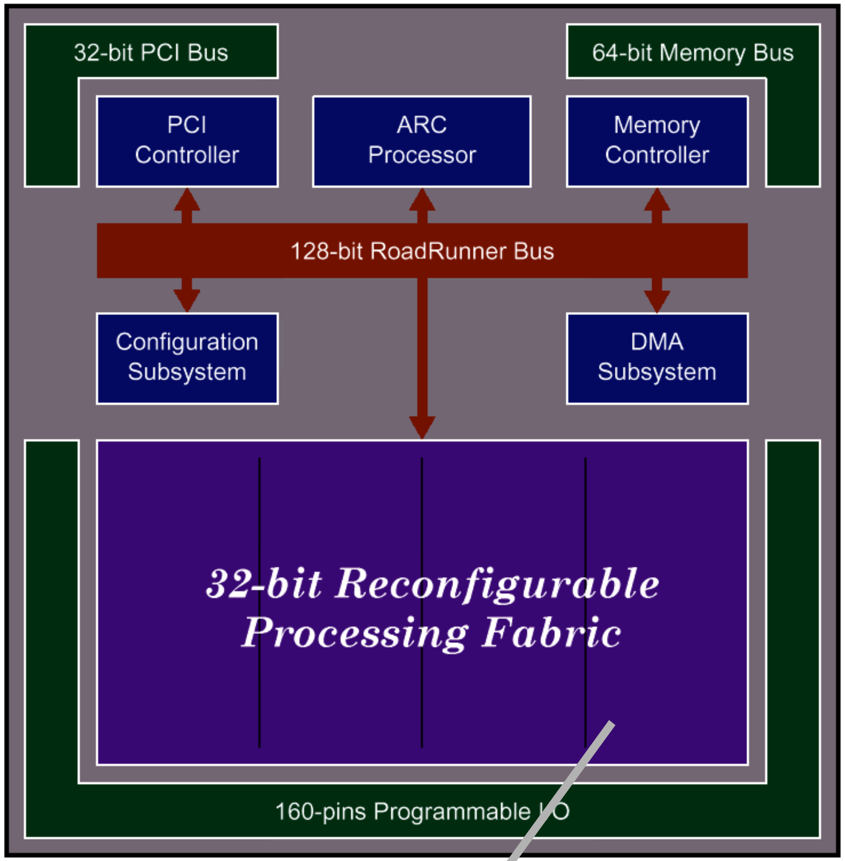


Source: Tensilica, Inc

Industrial Example 2: Chameleon RCP (Reconfigurable Communications Processor)



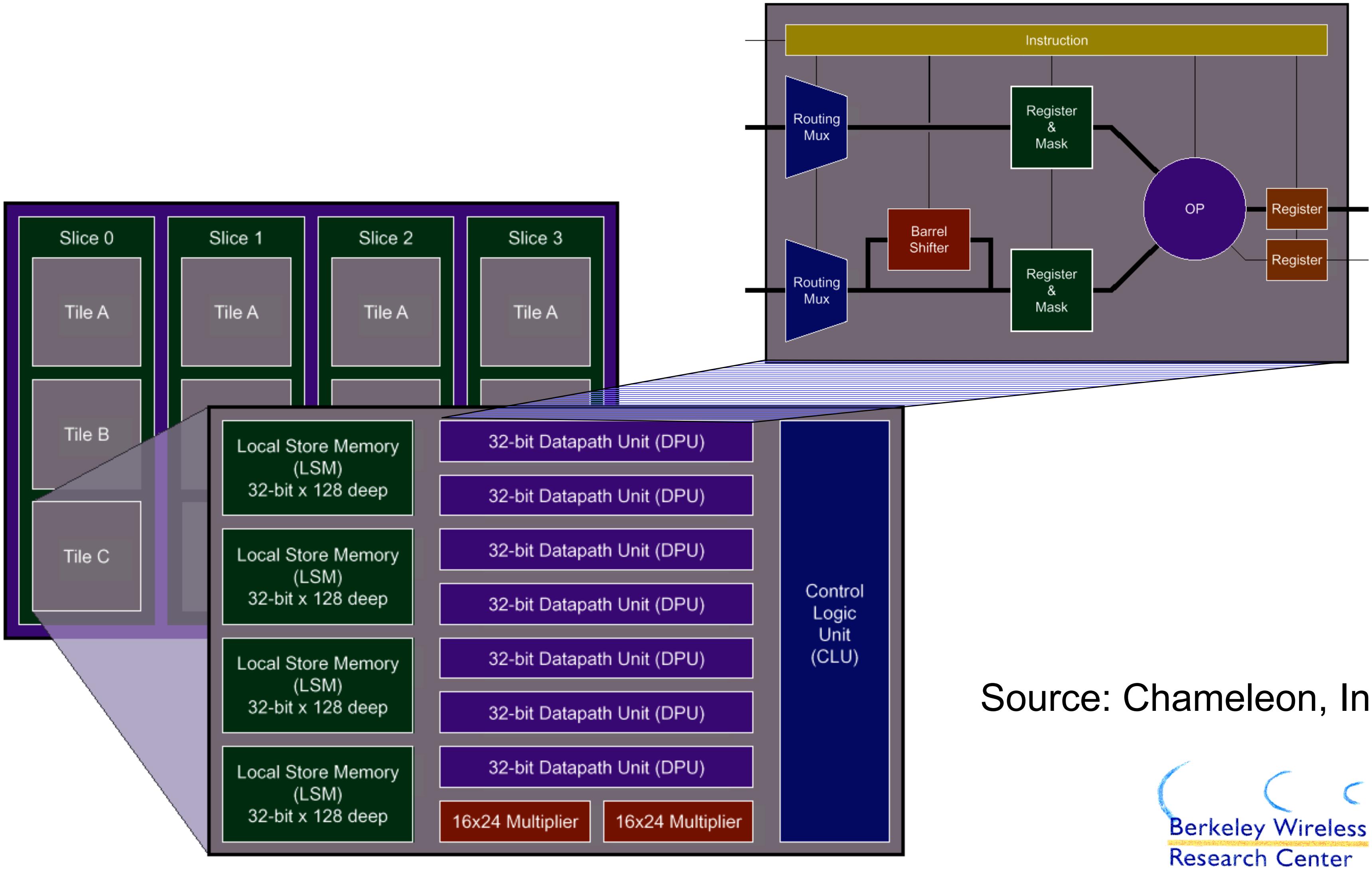
> Chameleon Systems is clearly plowing new ground with an instantaneously reconfigurable processor that is capable of performing high-end DSP functions that until now have been possible only through specialized ASICs and FPGAs. The CS2000 will get a lot of interest in hot new DSP markets, like 3G wireless basestations.
>
> — *Will Strauss, Forward Concepts*



24 multipliers
128 DPUs

Source: Chameleon, Inc

Reconfigurable Processing Fabric

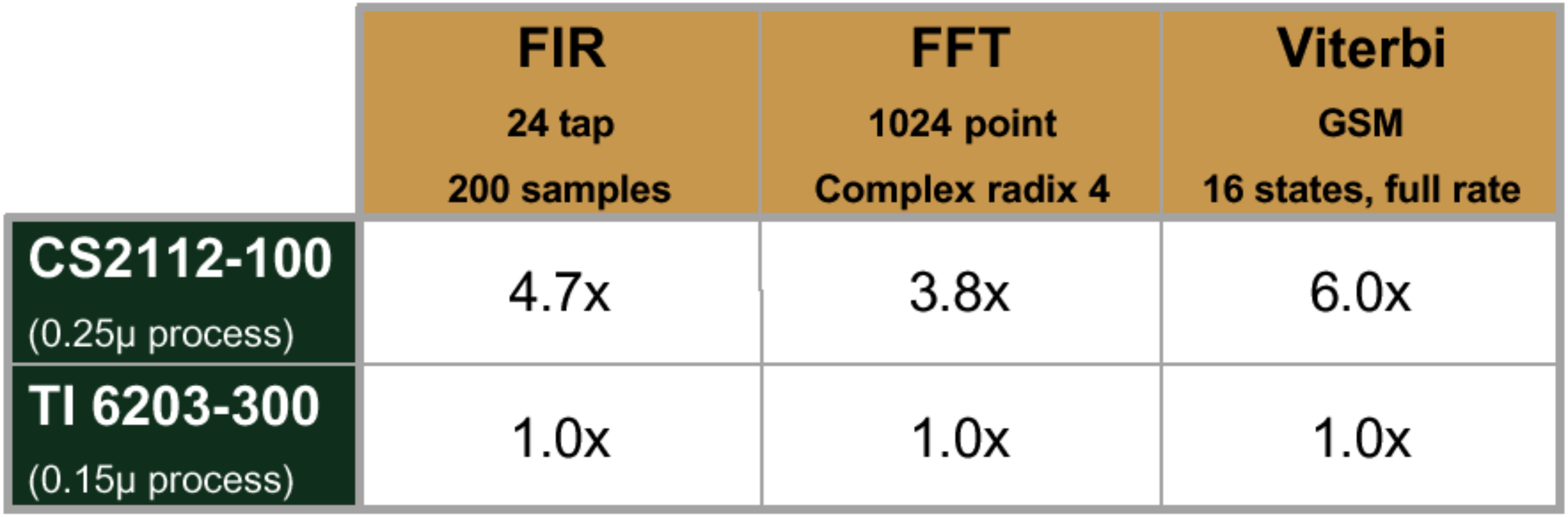

Instruction
Routing Mux
Routing Mux
Register & Mask
Register & Mask
Barrel Shifter
OP
Register
Register
Slice 0
Slice 1
Slice 2
Slice 3
Tile A
Tile A
Tile A
Tile A
Tile B
Tile C
Local Store Memory (LSM) 32-bit x 128 deep
Local Store Memory (LSM) 32-bit x 128 deep
Local Store Memory (LSM) 32-bit x 128 deep
Local Store Memory (LSM) 32-bit x 128 deep
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
32-bit Datapath Unit (DPU)
16x24 Multiplier
16x24 Multiplier
Control Logic Unit (CLU)

Source: Chameleon, Inc

CS2000 Performance Numbers

16-bit Operations/sec (w/o shift ops)	24,000 M
16-bit Operations/sec (w/shift ops)	45,000 M
16-bit MAC/sec	3,000 M
On-Chip Memory Bandwidth	48 GByte/sec
Main Memory Bandwidth	1 GByte/sec
Programmable I/O Bandwidth	1 GByte/sec
1024-point FFT	10 μsec
48-tap Symmetric FIR	125 MSample/sec
cdma2000 Chip Rate Processing	50 channels
UMTS Chip Rate Processing	16 channels
Viterbi	200 channels

CS2000 Performance

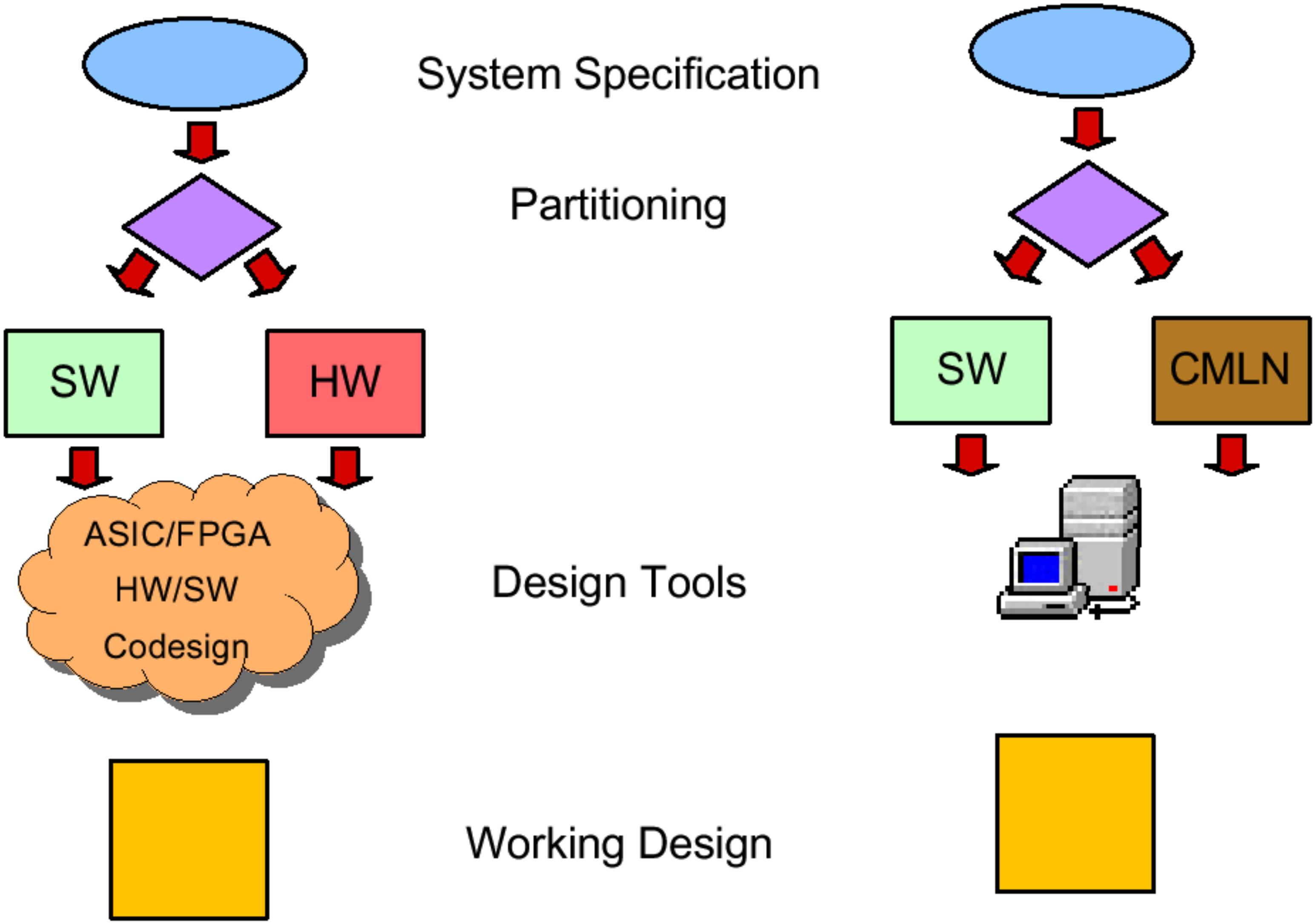


	FIR 24 tap 200 samples	FFT 1024 point Complex radix 4	Viterbi GSM 16 states, full rate
CS2112-100 (0.25µ process)	4.7x	3.8x	6.0x
TI 6203-300 (0.15µ process)	1.0x	1.0x	1.0x

Source: Chameleon, Inc

Power efficiency?

Design Methodology



System Specification

Partitioning

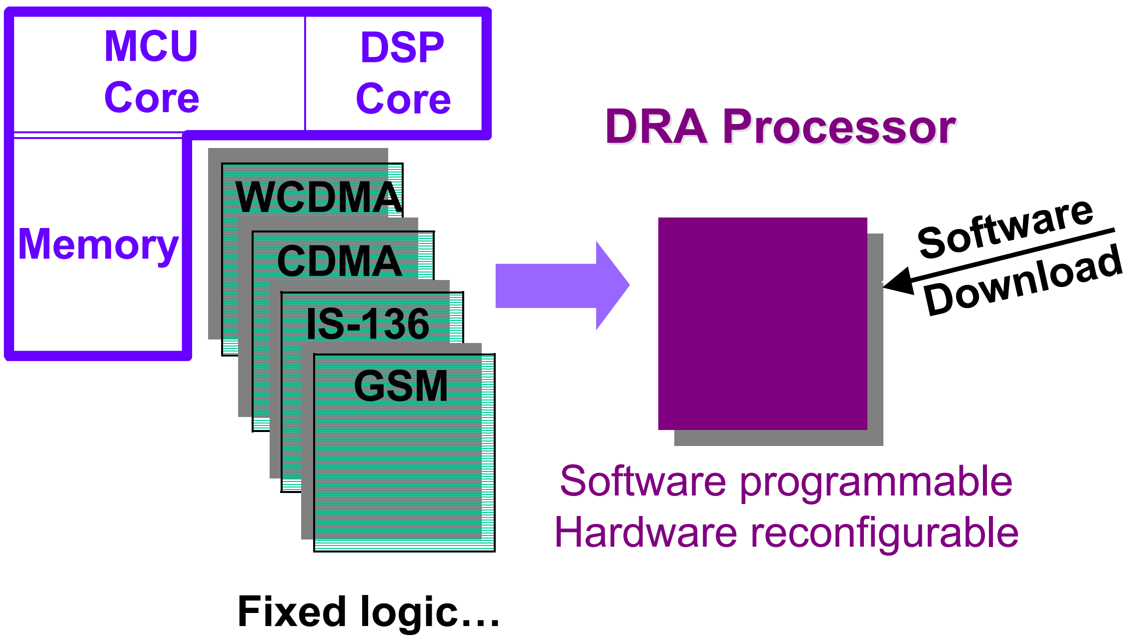
Design Tools

Working Design

Source: Chameleon, Inc

Industrial Example 3:

The MorphICs Dynamically Reconfigurable Architecture (DRA)



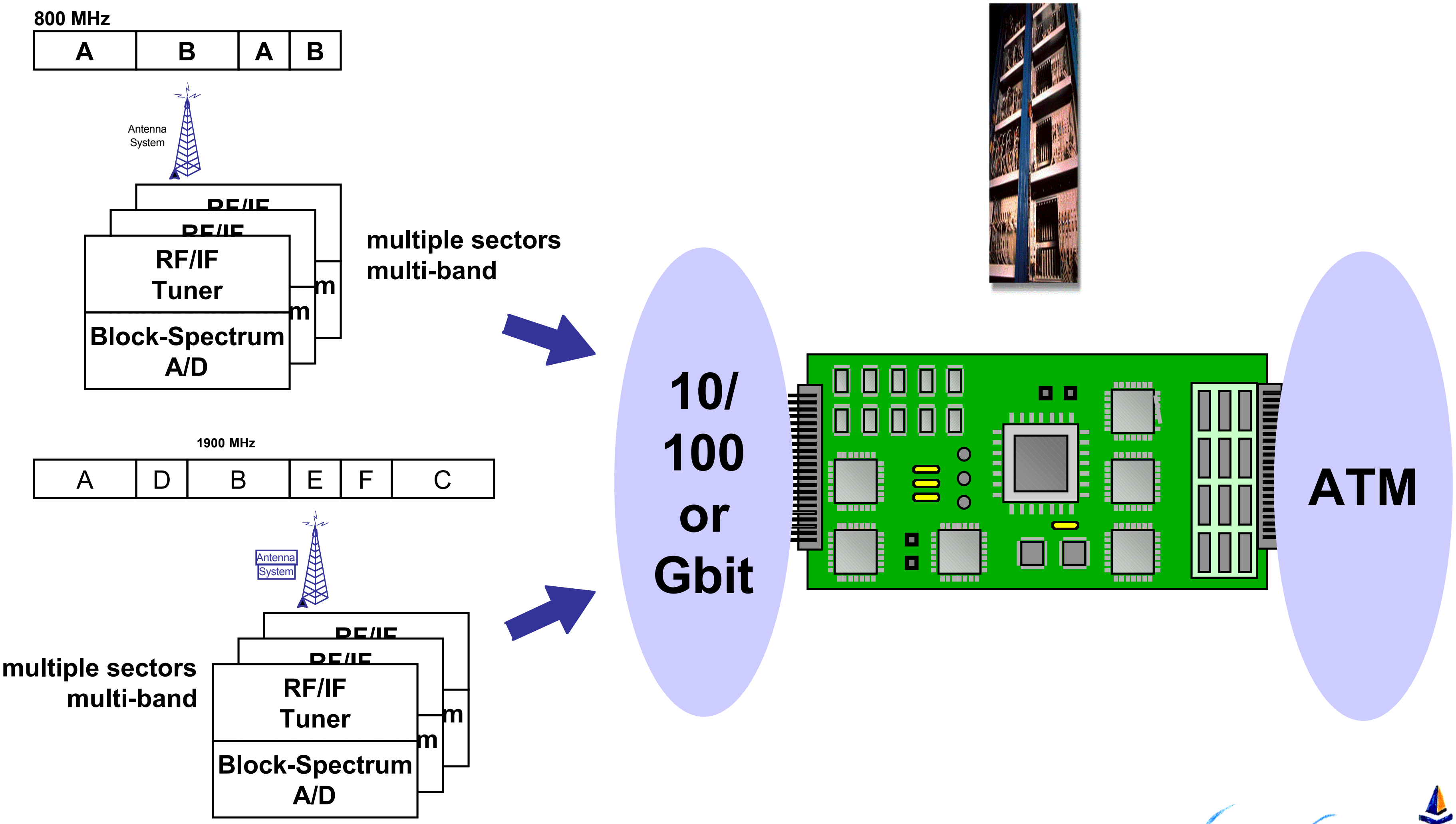
WCDMA	(mode, param)
CDMA	(mode, param)
WTDMA	(mode, param)
TDMA	(mode, param)

- SIM Card
- Handset Memory
- POS Programming
- Network Download
- OTA Download

Realizes cost, size and power targets similar to traditional core+hardwired

Source: Morphics Technology

Basestation of the Next Generation Wireless



800 MHz

A	B	A	B

Antenna System

RF/IF
RF/IF
RF/IF Tuner
Block-Spectrum A/D
m
m

multiple sectors
multi-band

1900 MHz

A	D	B	E	F	C

Antenna System

RF/IF
RF/IF
RF/IF Tuner
Block-Spectrum A/D
m
m

multiple sectors
multi-band

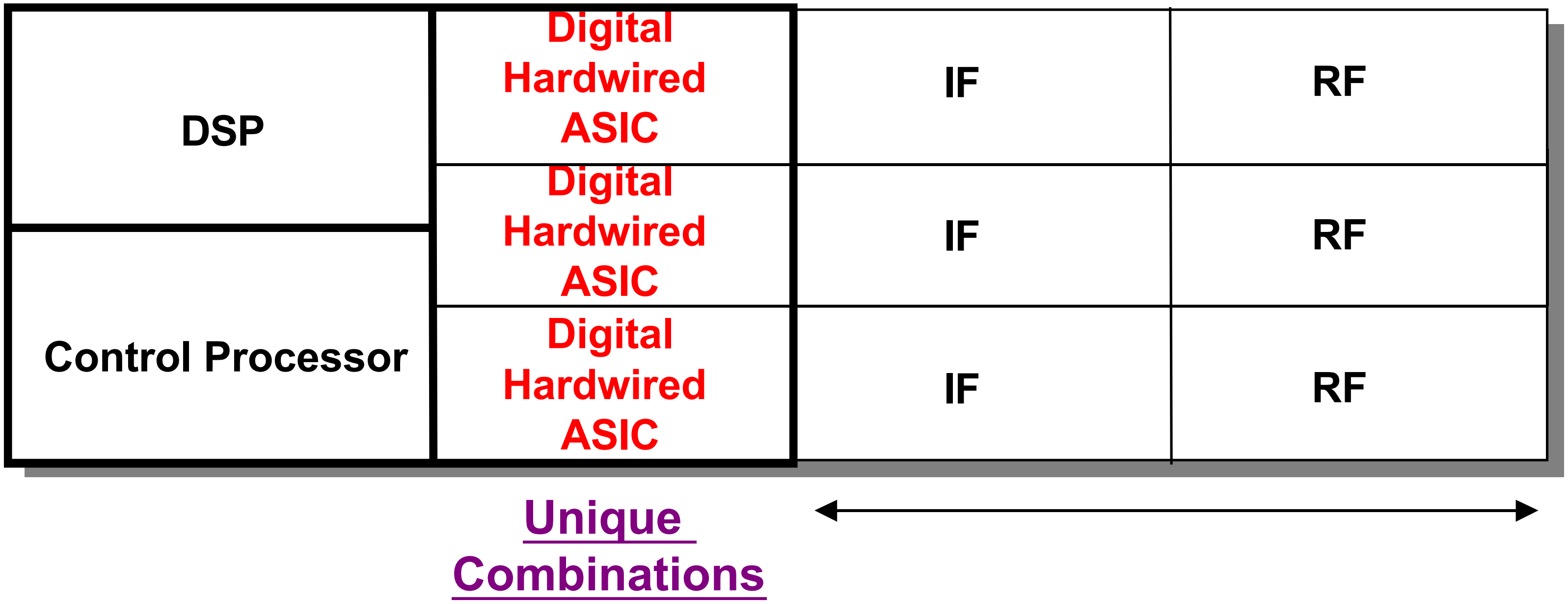
10/
100
or
Gbit

ATM

HW Multistandard Solutions

The common approach to hardware design involves:

multiple ASIC's to support each standard.



DSP

Control Processor

Digital Hardwired ASIC | IF | RF

Digital Hardwired ASIC | IF | RF

Digital Hardwired ASIC | IF | RF

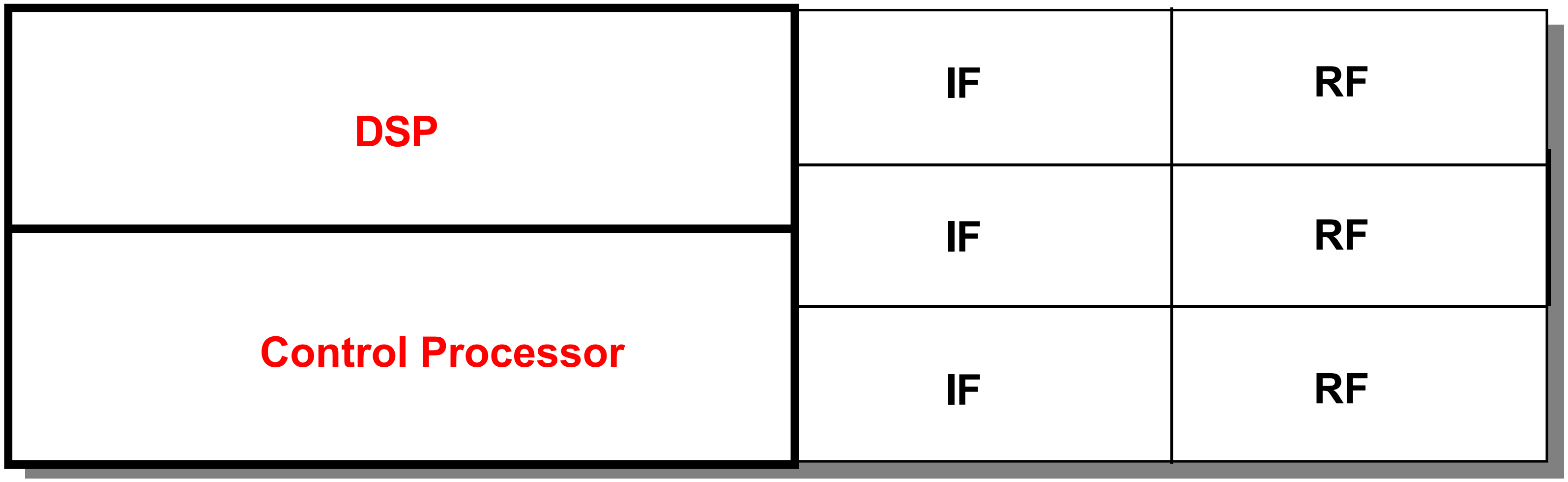
Unique Combinations

Hardwired implementation is not scalable or upgradeable to new standards.

- This approach costs time in a time-to-market dominated world.
- Creating new chipsets for every technology combination critically challenges available design resources!

SW Multistandard Solution

Applying instruction-set processor architectures to all baseband processing would be desireable...



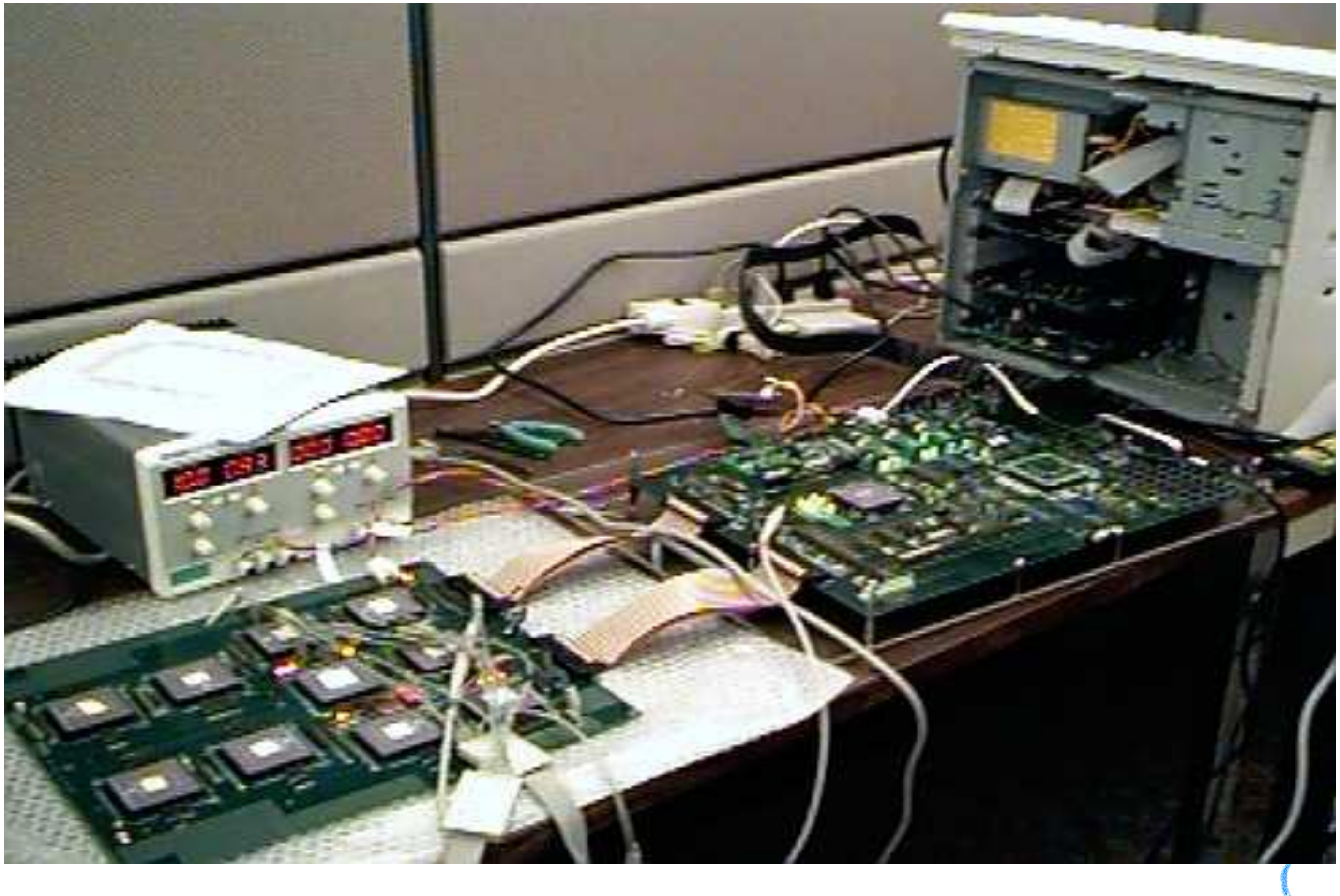
but is simply not an good implementation for base stations:

-Unacceptably high cost per channel

-Unacceptably large power per channel

This is definitely not a viable implementation for terminals

FPGA the Solution?
Cellular Handset Using Current FPGA



Source: Morphics Technology

Berkeley Wireless Research Center

Successfully Using Reconfigurability

Application-Specific Leverage

Focus on first on applications and constituent algorithms, not the silicon architecture !

Wireless Communications Transceiver Signal Processing

Minimize **the hardware reconfigurability to constrained set**

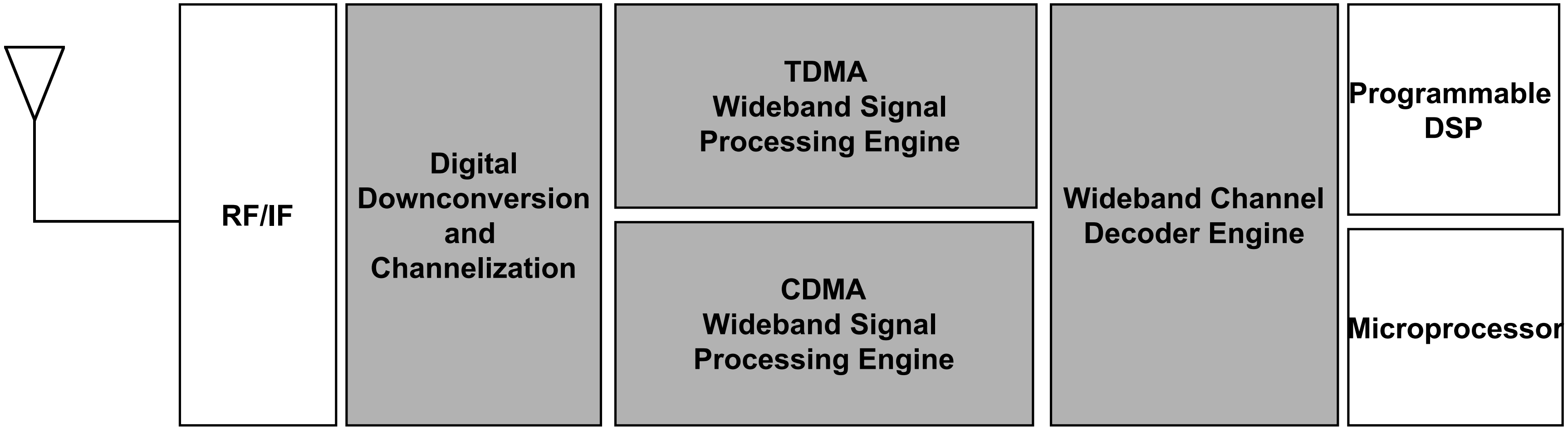
Maximize **the software parameterizability and ease of use of the programmer's model for flexibility**

➡ Define **optimal architecture for efficient implementation**

Source: Morphics Technology

Berkeley Wireless Research Center

Application-Specific MOPS in Digital Communications



RF/IF

Digital Downconversion and Channelization

TDMA Wideband Signal Processing Engine

CDMA Wideband Signal Processing Engine

Wideband Channel Decoder Engine

Programmable DSP

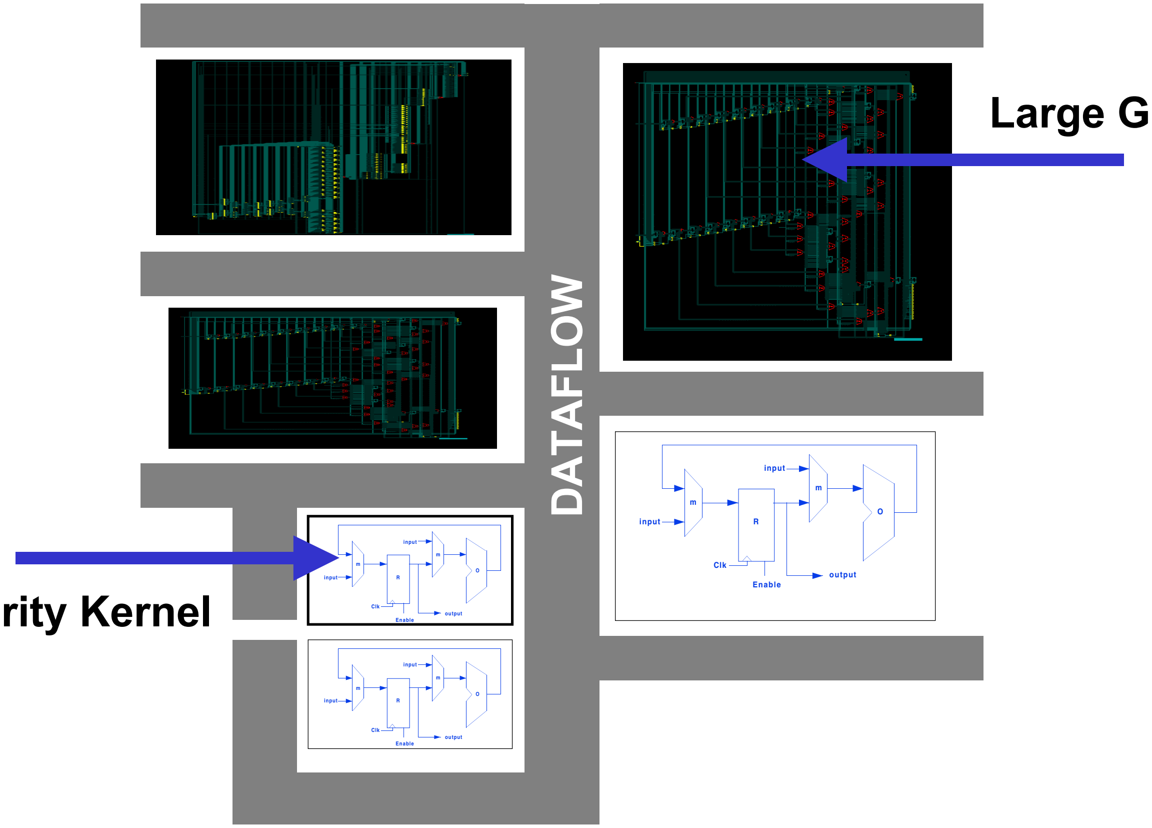
Microprocessor

Source: Morphics Technology

Berkeley Wireless Research Center

Morphics' DRL Architecture

Heterogeneous Multiprocessing Engine Using *Application-Specific Reconfigurable Logic*

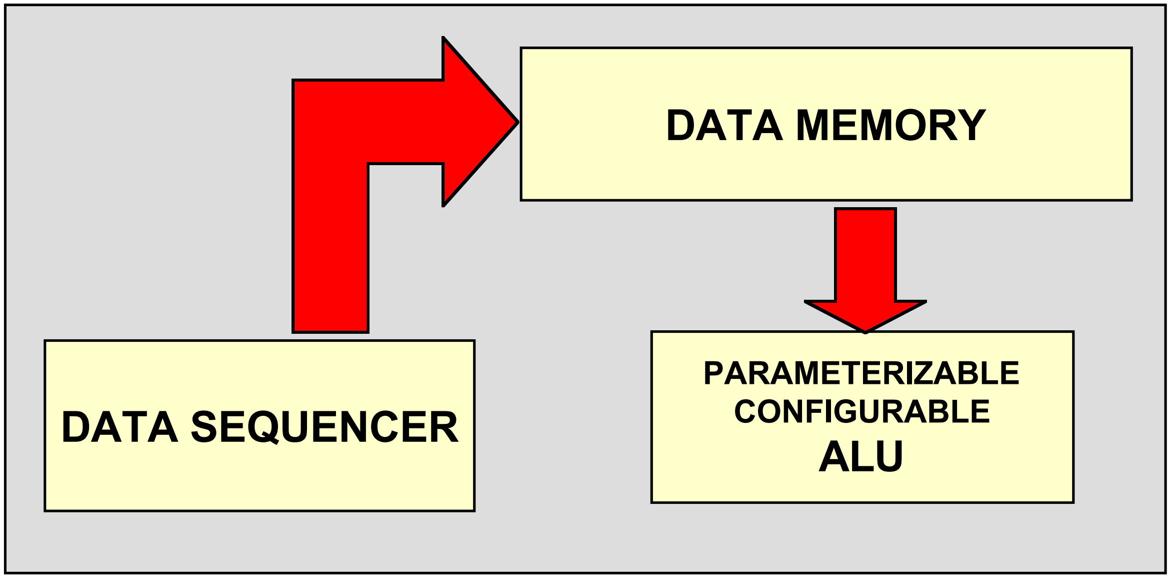


Large Granularity Kernel

Small Granularity Kernel

Source: Morphics Technology

DRL Kernels



Source: Morphics Technology

Key Pieces of Design Methodology

System-level Profiling

- Analyze sequences of operations (arithmetic, memory access, etc)
- Analyze communication bottlenecks
- Key flexible parameters (algorithm v architecture parameters)

Architecture-level Profiling

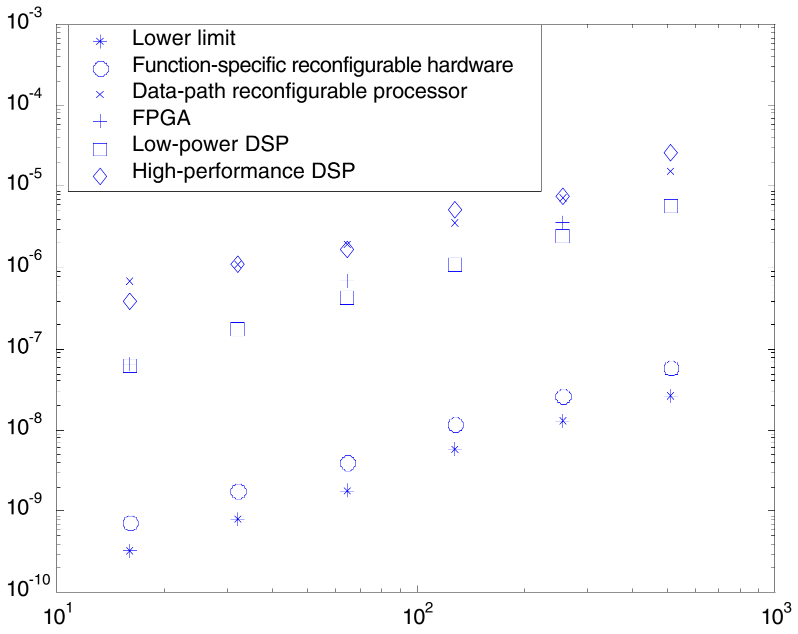
- ALU/kernel definition (sequences of operators)
- Memory profile
- Type of configurability required for flexibility
- Macro-sequencer development

Implementation

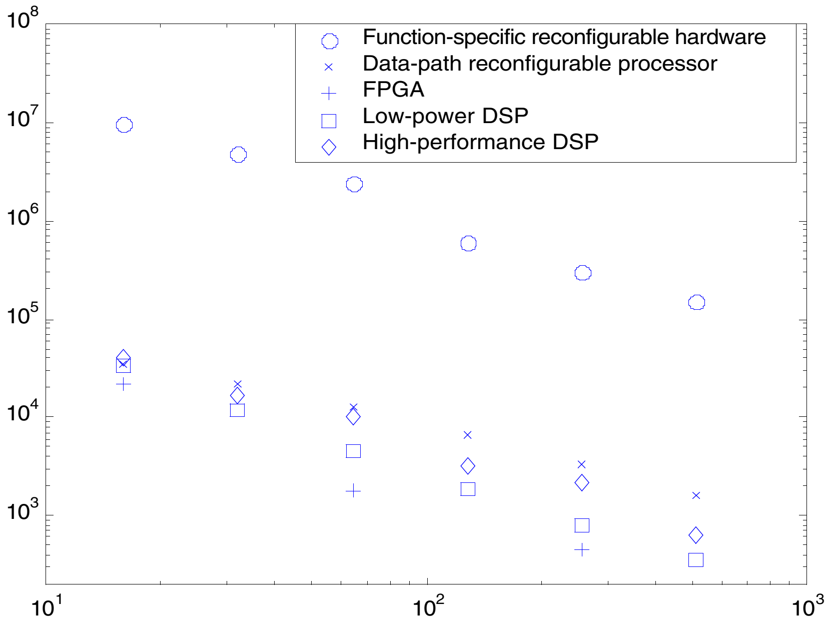
- SW- programmer’s model developed at architecture specification stage
- SW- API proven out via behavioral models & demonstrator hardware
- VLSI-focus on regular predictable timing and routability
- VLSI- embedded reconfigurability in an ASIC flow

Source: Morphics Technology

Architectural Implementation Comparison: Reconfigurable FFT



Energy per Transform
vs. FFT size

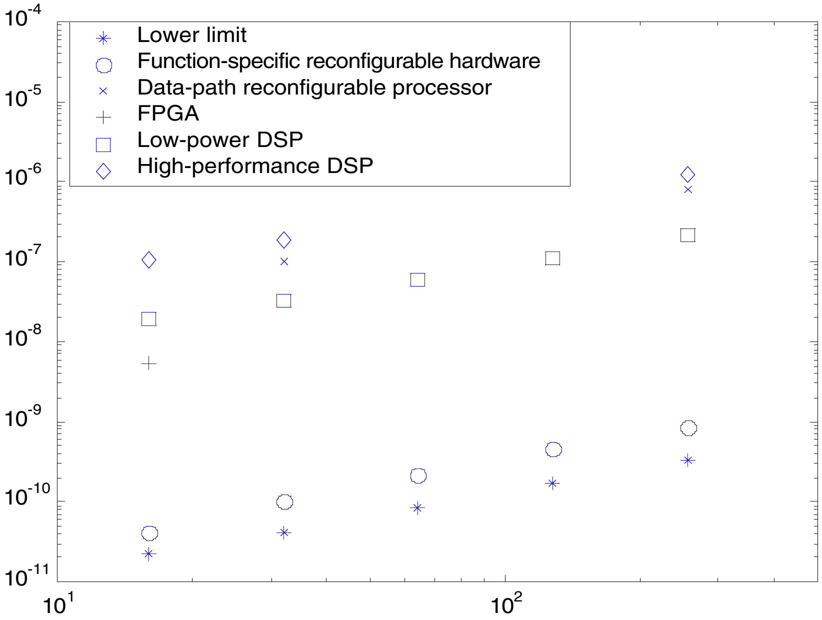


Transforms per Second per mm^2
vs. FFT size

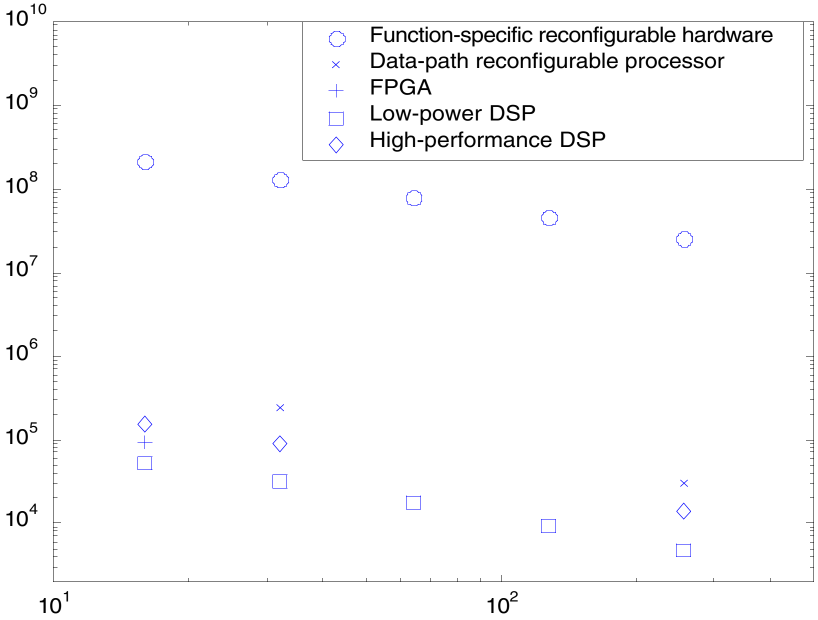
* All results are scaled to 0.18µm

Berkeley Wireless
Research Center

Architectural Implementation Comparison: Reconfigurable Viterbi Decoder



Energy per Decoded Bit vs. Number of States

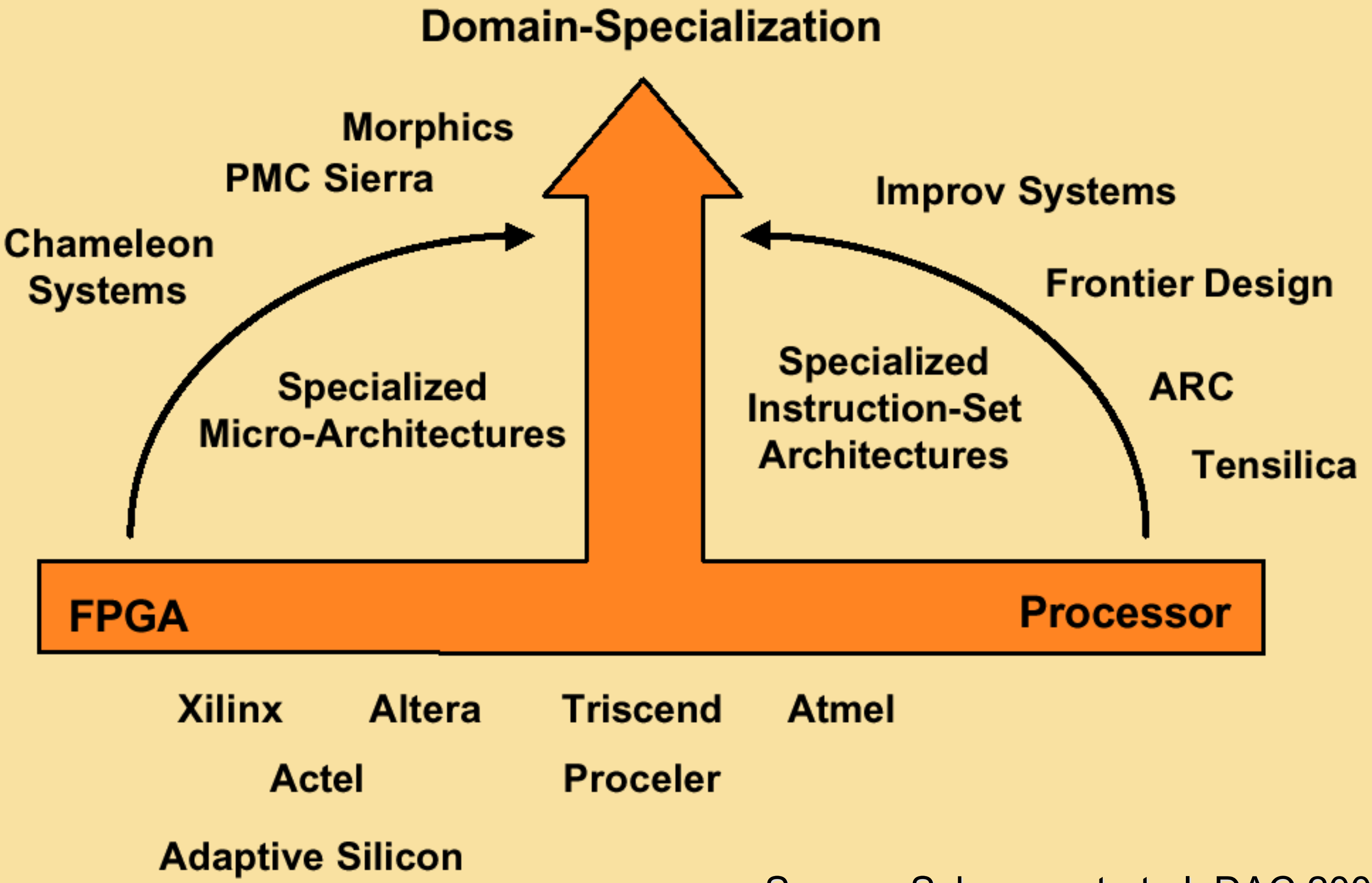


Decoding Rate per mm^2 vs. Number of States

* All results are scaled to 0.18μm

Berkeley Wireless Research Center

Commercial Specializations



Domain-Specialization

Morphics

PMC Sierra

Chameleon Systems

Specialized Micro-Architectures

Improv Systems

Frontier Design

Specialized Instruction-Set Architectures

ARC

Tensilica

FPGA

Processor

Xilinx

Altera

Triscend

Atmel

Actel

Proceler

Adaptive Silicon

Source: Schaumont et al, DAC 2001

Summary

- **Configurable computing is finding its way into the embedded processor space**
- **Best suited (so far) for**
  - Flexible I/O and Interface functionality
  - providing task-level acceleration of “parametizable” functions
- **Software flow still subject to improvement**

DO NOT FORGET CONFIGURATION OVERHEAD