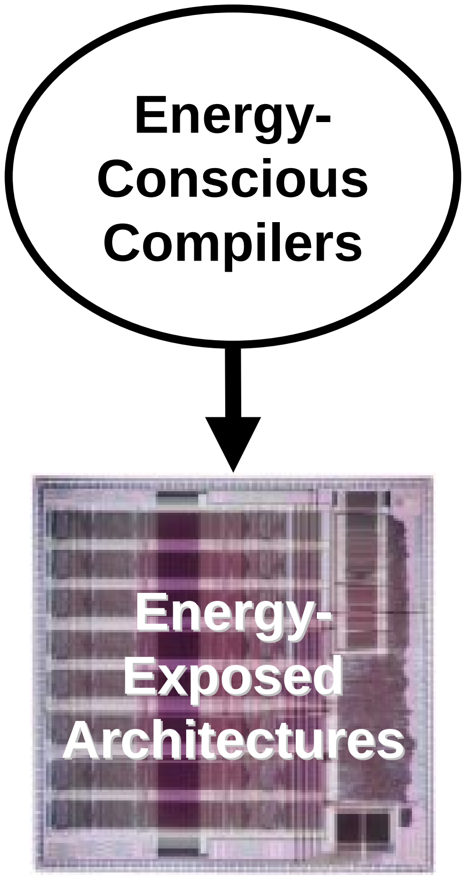
SCALE:
Software-Controlled Architectures for Low Energy

Krste Asanovic

MIT Laboratory for Computer Science

SCALE Project Goal

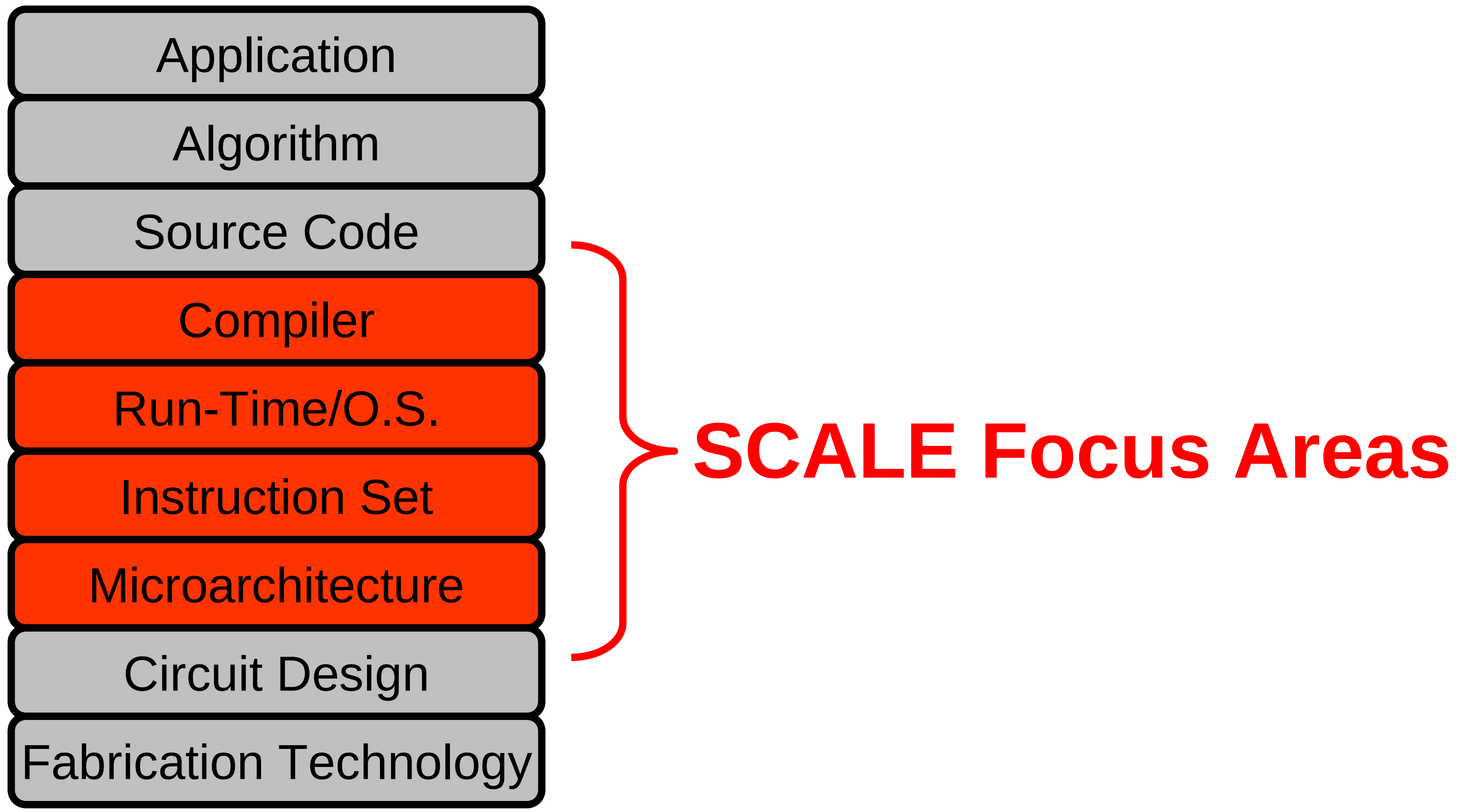
Improve Energy-Efficiency of Programmable Processors by Re-Examining Hardware-Software Interface



Motivation

- **Power dissipation limits many system designs**
  - battery weight and life for portable devices
  - packaging and cooling costs for tethered systems
  - case temperature for wearable computers (user comfort)
- **Custom circuits (ASICs) use least energy**
  - only for small, regular kernels
  - only feasible for high volume applications
  - cannot adapt to changing requirements
- **Programmable processors are most flexible**
  - single design reusable in many systems
  - can change application code after fabrication
  - ***but up to 100-1000x more energy dissipation than ASIC***

Can Optimize Energy Efficiency at all Levels in System



Improving Energy-Efficiency at Compiler and Architecture Levels

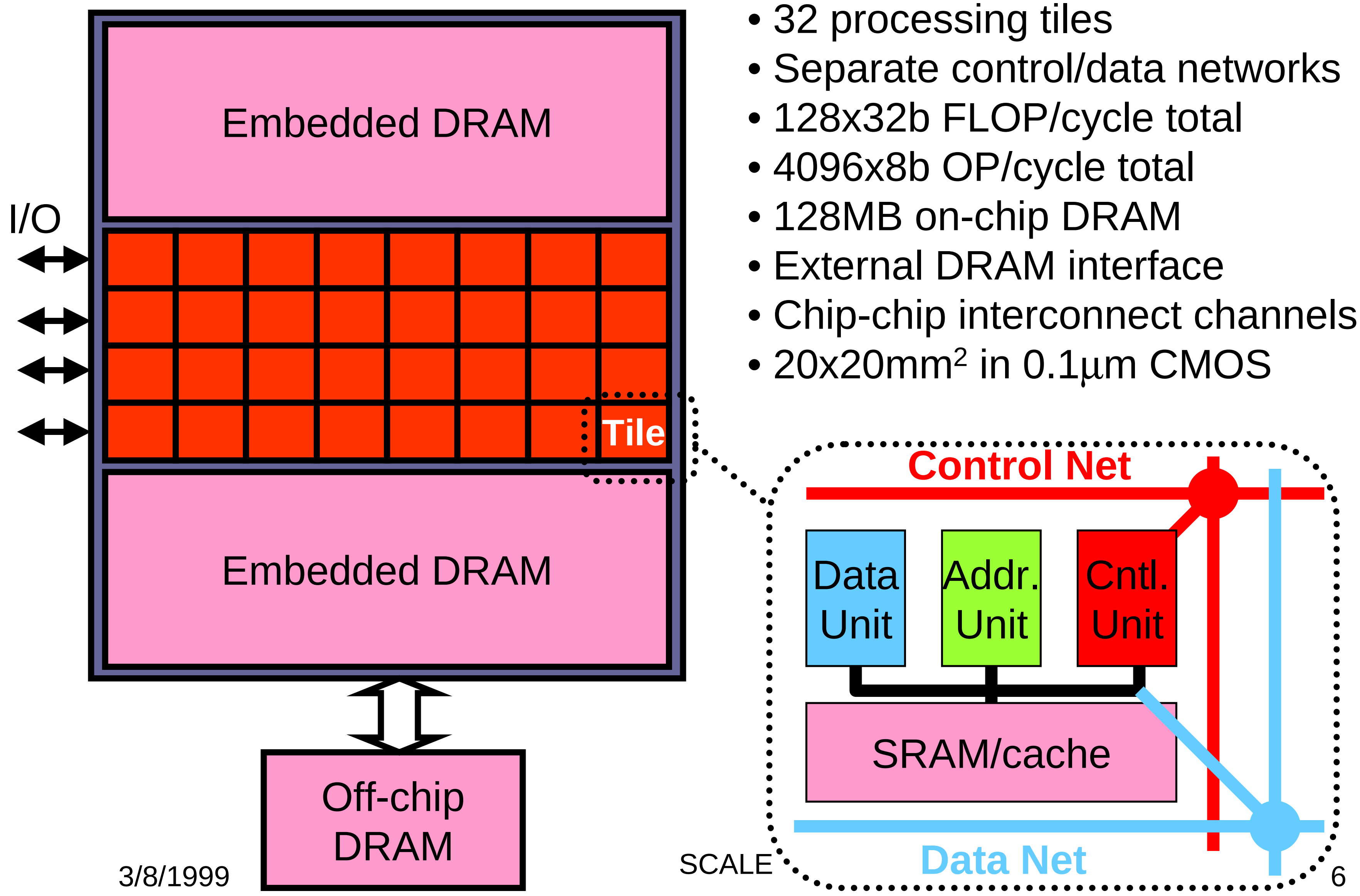
- **Increase performance**

(voltage scaling trades excess performance for lower energy)

  - highly parallel machine architectures
  - aggressive compiler optimizations
  - hardware support for common compute paradigms

- **Reduce unnecessary switching activity**
  - power down unneeded units
  - reduce datapath widths to avoid excess precision
  - localize computations to minimize data communication
  - configure control to minimize control overhead

SCALE Processor Overview



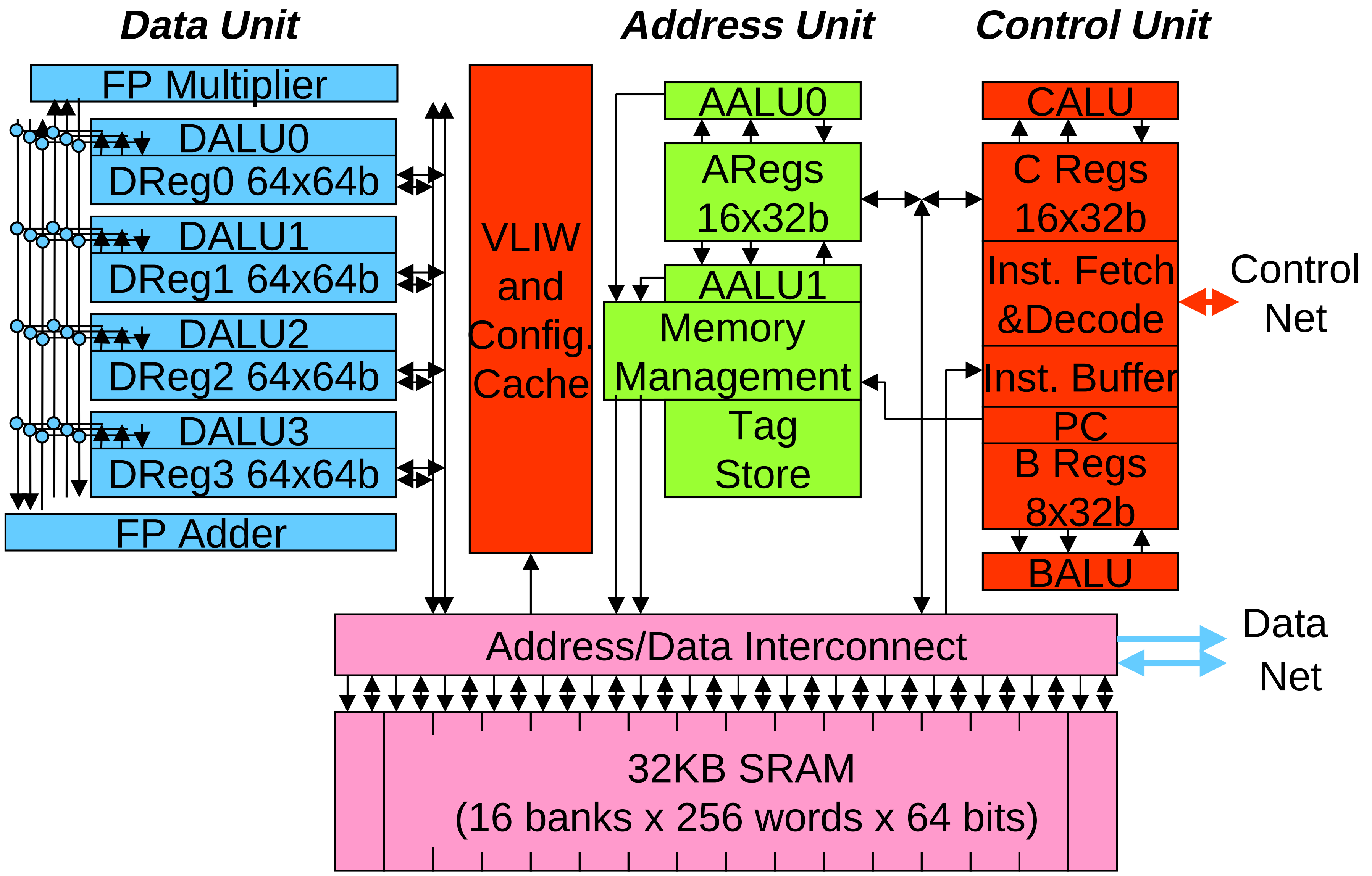
- 32 processing tiles
- Separate control/data networks
- 128x32b FLOP/cycle total
- 4096x8b OP/cycle total
- 128MB on-chip DRAM
- External DRAM interface
- Chip-chip interconnect channels
- 20x20mm^2 in 0.1µm CMOS

SCALE Processor Supports All Forms of Parallelism

- Multithreaded/Chip-scale multiprocessor
  - Run separate threads on different tiles
- Vector
  - Hardware control for vector arithmetic and vector memory operations
- VLIW/Reconfigurable
  - Distributed wide instruction cache/configuration buffer allows software to drive exposed datapath control lines

Control net can lock multiple tiles together for greater single thread performance in vector or VLIW mode

SCALE Processor Tile Details



Software Power Control

SCALE processor has extensive software-controlled power down capability

- Turn off unused register banks and ALUs in each unit
- Reduce datapath width
  - set width separately for each unit in tile (e.g., 32b in control unit, 16b in address unit, 64b in data unit)
- Turn off individual local memory banks
- Turn off idle tiles and idle inter-tile network segments
- Turn off refresh to unused DRAM banks

SCALE Exposes Locality at Multiple Levels

- **2D Tile and DRAM layout**
  - software maps computation to minimize network hops
- **Local SRAM within tile**
  - software split between instruction/data/unified storage
  - software scratchpad RAMs or hardware-managed caches
- **Distributed cached control state within tile**
  - control unit: instruction buffer
  - data/address unit: vector instructions or VLIW/configuration cache
- **Distributed regfile and ALU clusters within tile**
  - Control Unit: scalar (C) registers versus branch (B) registers
  - Address Unit: address (A) registers
  - Data Unit: Four clusters of data registers (D0-D4)
  - Accumulators and sneak paths to bypass register files

Backup Slides

SCALE Tile Resources

- Control Unit
  - instruction fetch/decode, branch & loop execution
  - scalar integer compute
  - control flow synchronization with other tiles over control net
- Address Unit
  - address generation and address mapping
  - local memory and cache management
  - global memory accesses over data net
- Data Unit
  - floating-point and fixed-point computation
  - 64-bit datapath configurable as 2x32-bit, 4x16-bit, 8x8-bit
  - large register file (256x64-bit elements)
- Local Memory
  - 16 banks x 2KBytes/bank (total 32KBytes SRAM)
  - 128 Bytes/cycle peak bandwidth
  - configurable as scratchpad RAM or cache

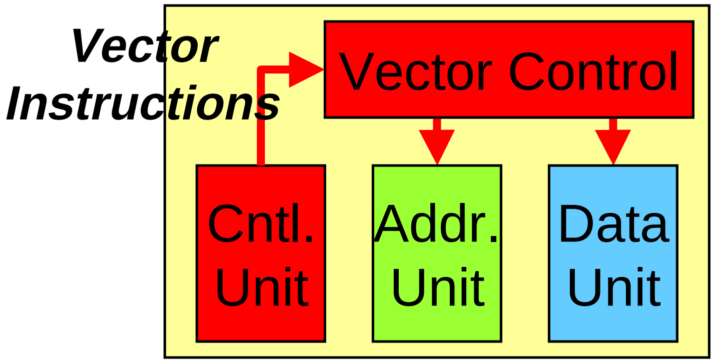
SCALE Supports All Forms of Parallelism

Vector

– most streaming applications highly vectorizable
– vectors reduce instruction fetch/decode energy up to 20-60x (depends on vector length)
– mature programming and compilation model

⇒***SCALE supports vectors in hardware***

– address and data units optimized for vectors
– hardware vector control logic

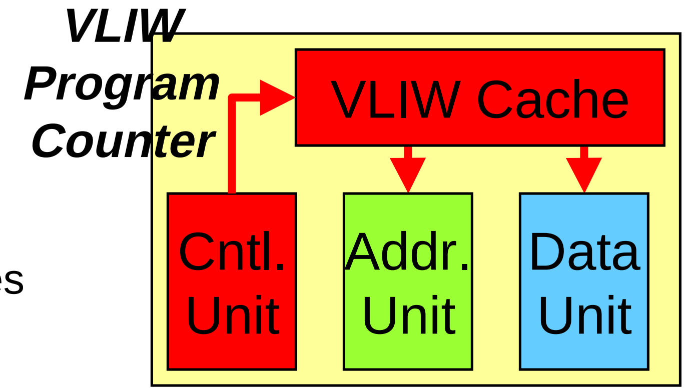


VLIW/Reconfigurable

– exploit instruction-level parallelism for non-vectorizable applications
– superscalar ILP expensive in hardware

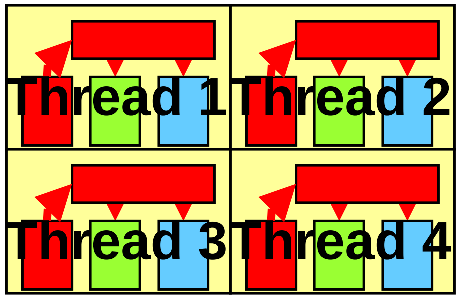
⇒***SCALE supports VLIW-style ILP***

– reuse address and data unit datapath resources
– expose datapath control lines
– single wide instruction = configuration
– provide control/configuration cache distributed along datapaths

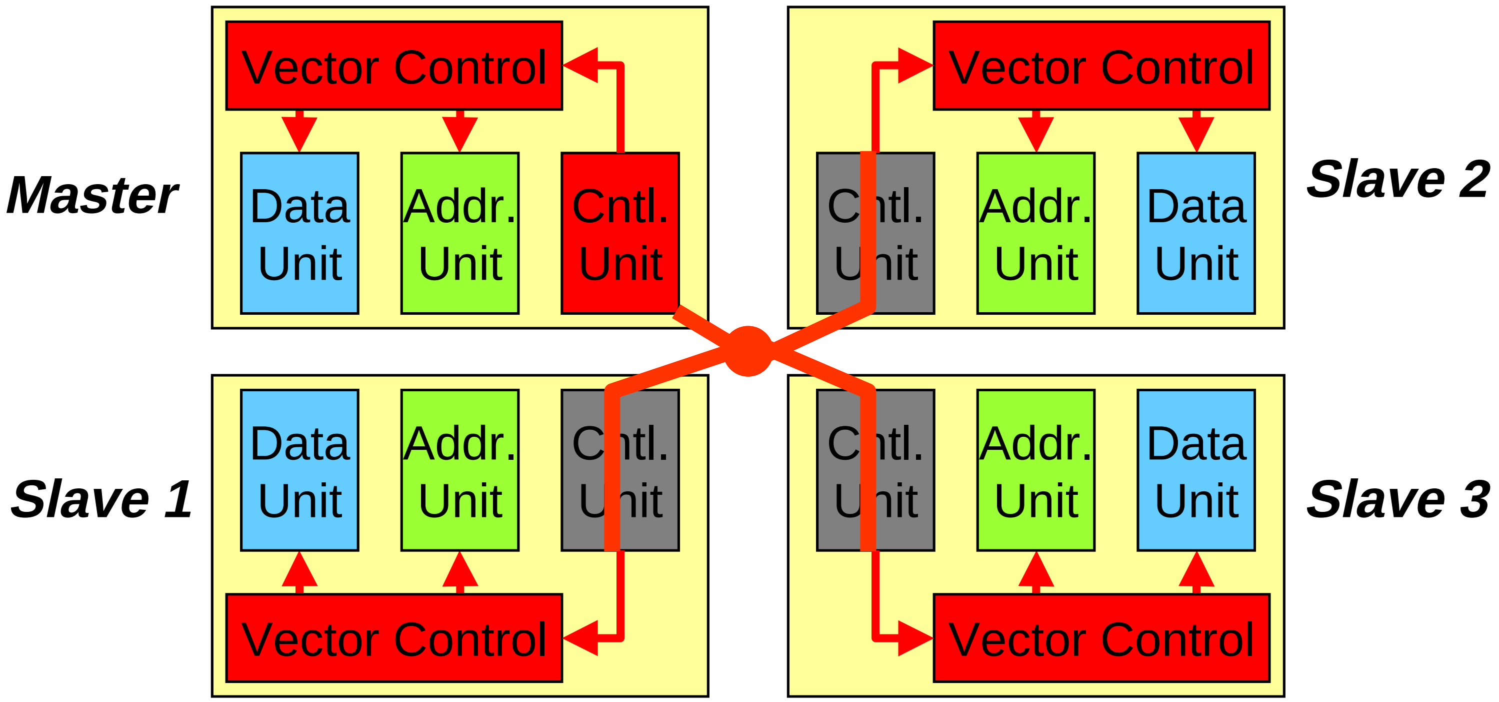


Multithreading/Chip-scale Multiprocessor

– run separate threads on different tiles
– any mix of vector or VLIW across tiles



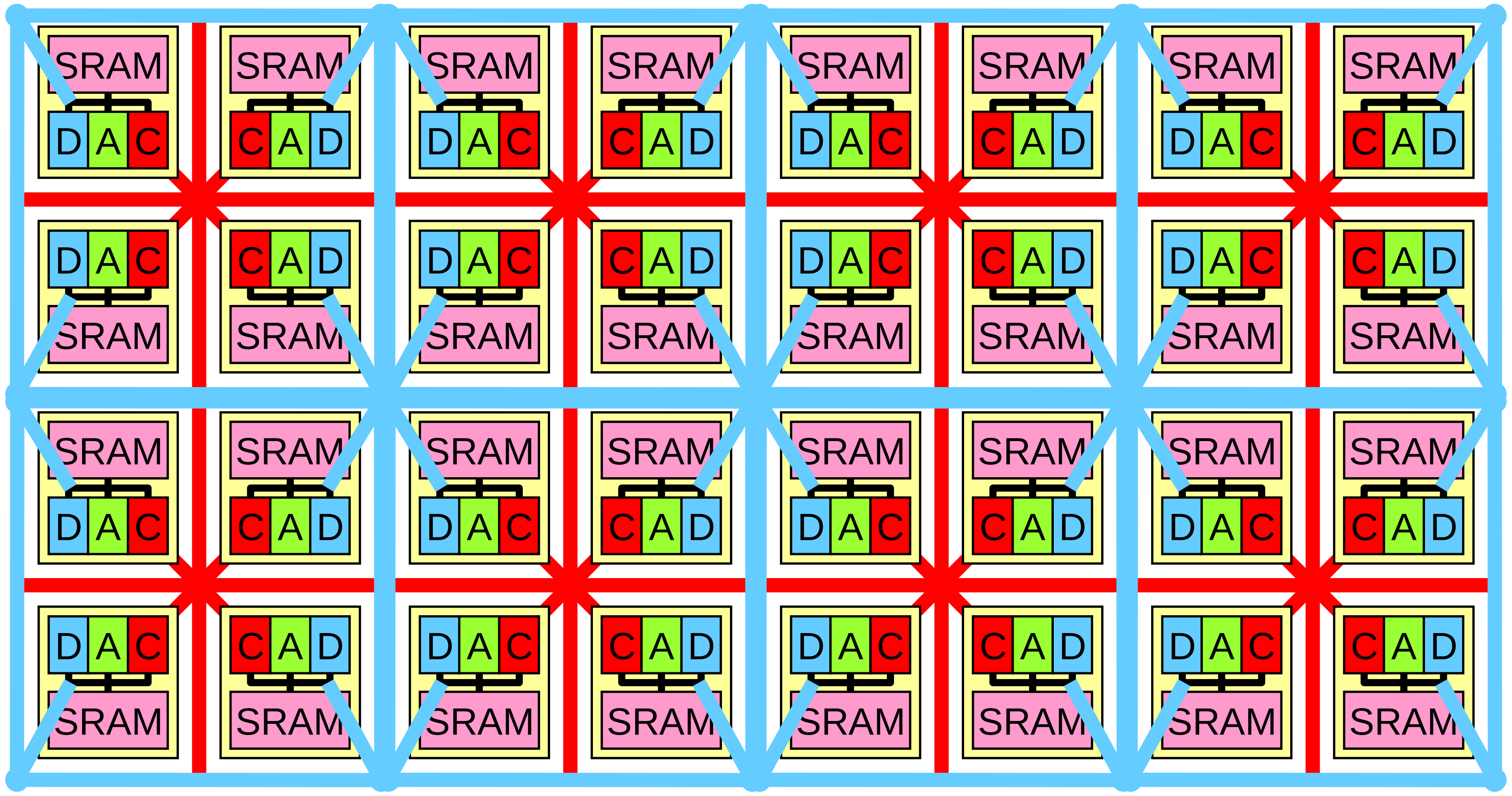
Tile Locking



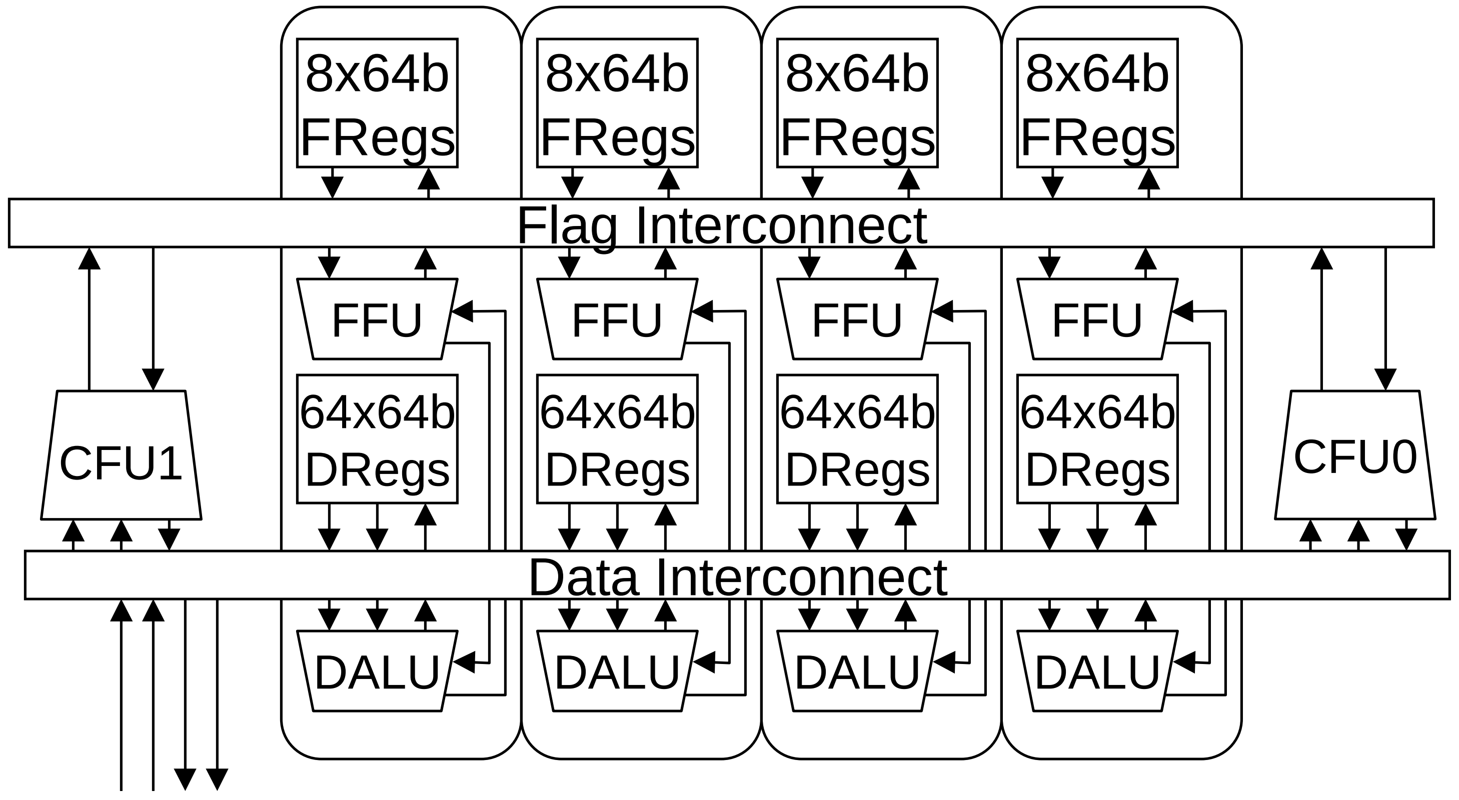
Lock slave tiles to master tile over control network

- increase single thread performance for vector and/or VLIW tiles
- amortize instruction fetch/decode energy over more datapaths
- amortize instruction storage over more tiles
- avoid overhead of software tile synchronization

SCALE Tile Array


SRAM SRAM SRAM SRAM SRAM SRAM SRAM SRAM
D A C C A D D A C C A D D A C C A D D A C C A D
D A C C A D D A C C A D D A C C A D D A C C A D
SRAM SRAM SRAM SRAM SRAM SRAM SRAM SRAM
SRAM SRAM SRAM SRAM SRAM SRAM SRAM SRAM
D A C C A D D A C C A D D A C C A D D A C C A D
D A C C A D D A C C A D D A C C A D D A C C A D
SRAM SRAM SRAM SRAM SRAM SRAM SRAM SRAM

SCALE Data Unit Structure



To Memory System/Tile Interconnect

Conventional Instruction Sets Hide Energy Consumption from Software

- RISC/VLIW instruction set architectures (ISAs) designed for high performance and simplicity

$\Rightarrow$ *ISA only provides alternative mechanisms when there is a potential **performance** gain*

Implicit assumption:

software only interested in performance

SCALE Philosophy

Reward compile-time knowledge with run-time energy savings

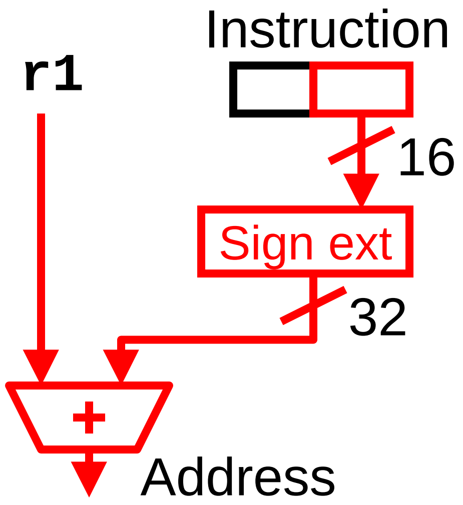
- hardware must provide alternative mechanisms which reduce energy (performance unchanged)
- software must be able to map computations to use lowest-energy hardware mechanisms

$\Rightarrow$ ***Co-develop energy-exposed architectures and energy-conscious compilers***

Example 1: Addressing Modes

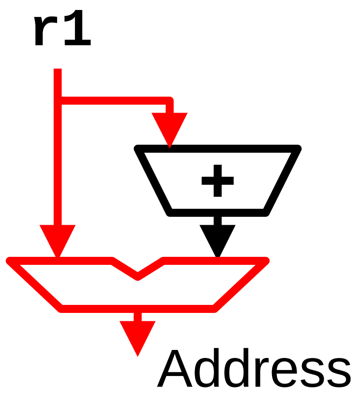
Conventional RISC,
only base+offset

`ld r1,0(r2)`



Alternate address mode,
pure register indirect

`ld r1,(r2)`



No immediate fetch, sign extension, or adder energy but extra multiplexor

Example 2: Branch Address

Conventional RISC

```
loop: ld r1,0(r2)
      add r2, #1
      bnez r1, loop
```

- Branch target address recalculated every time around loop

Explicit branch target register

```
      la btr, loop
loop: ld r1,(r2)
      add r2, #1
      bnez r1, (btr)
```

- Branch target address calculation moved out of loop *(no immediate fetch or add)*

Example 3: Tag-Unchecked Loads

Allow software to avoid cache tag check when successive memory accesses are to same cache line

`ld r1,(r2)`

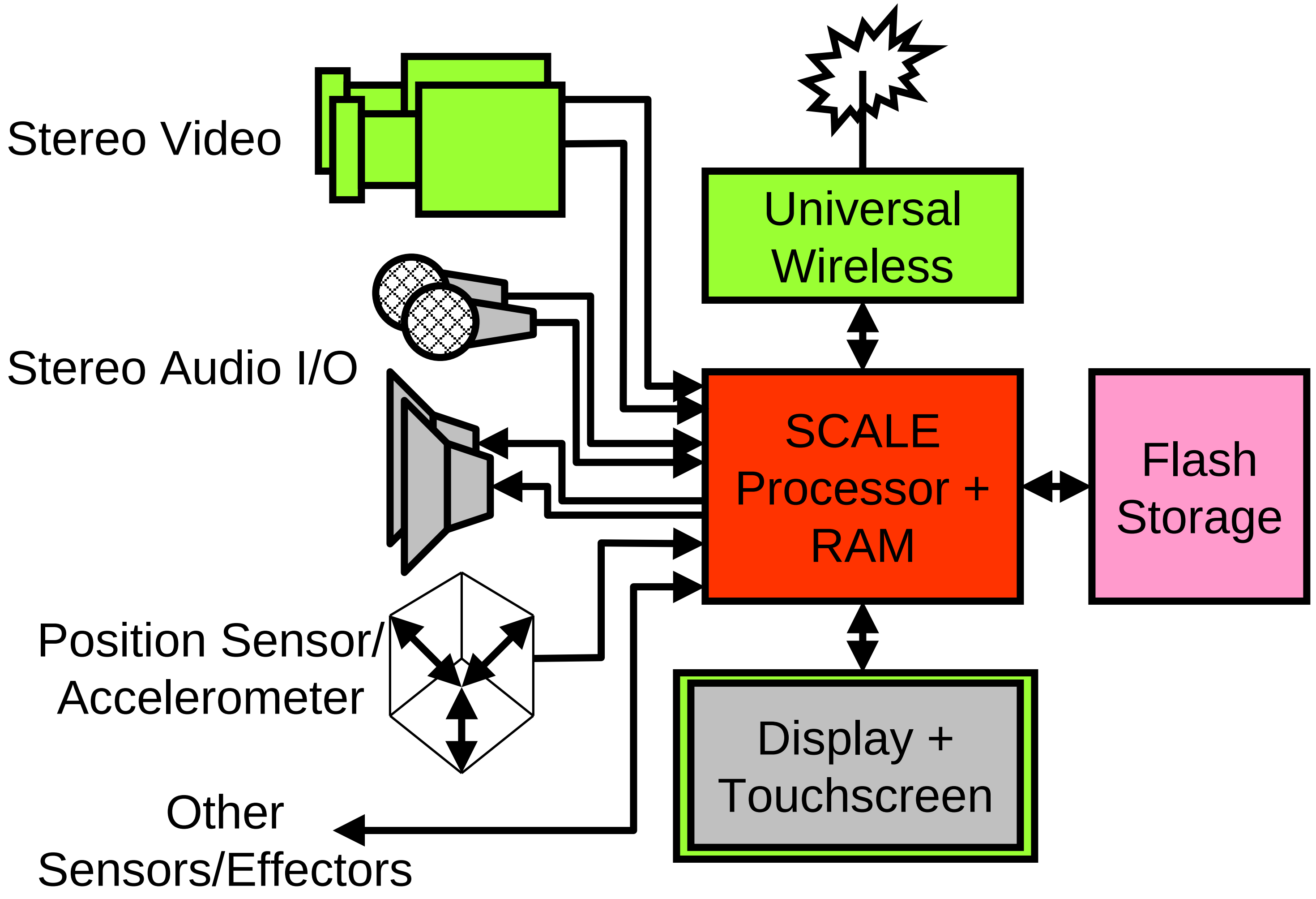
`ld.nochk r3,4(r2)` ⇐ *Must be to same cache line*

Energy reductions:

- no tag RAM read
- no tag compare
- only low order address bits need to be computed
- no TLB lookup

⇒ ***Reduces cache access energy to just RAM read***

SCALE Demonstration System



H21: A 21st Century Universal Handheld for Oxygen

Importance of General-Purpose Processing

- Not all code suitable for mapping to custom circuitry
  - Most ASICs include a GPP to handle complex code
- Amdahl’s law applied to energy consumption:
  - If 99% of an application moved to ASIC with 1000x energy reduction, remaining GPP will consume >90% final energy!

$\Rightarrow$ ***Must focus on complete applications***