Project ARIES

Advanced RAM Integration for Efficiency and Scalability

Presented by the 8th floor buttheads

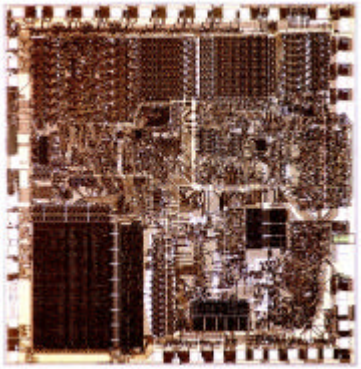
(pun intended)

Motivation - System Level

- Many applications have massive memory and/or computational requirements
  - graphics, CAD, physical simulation, factoring, etc.
- Current parallel systems have severe limitations:
  - difficult to program
  - poor network performance
  - insufficient scalability
- There is a need for programmable systems scalable to 1M processors and beyond

Motivation - Architecture

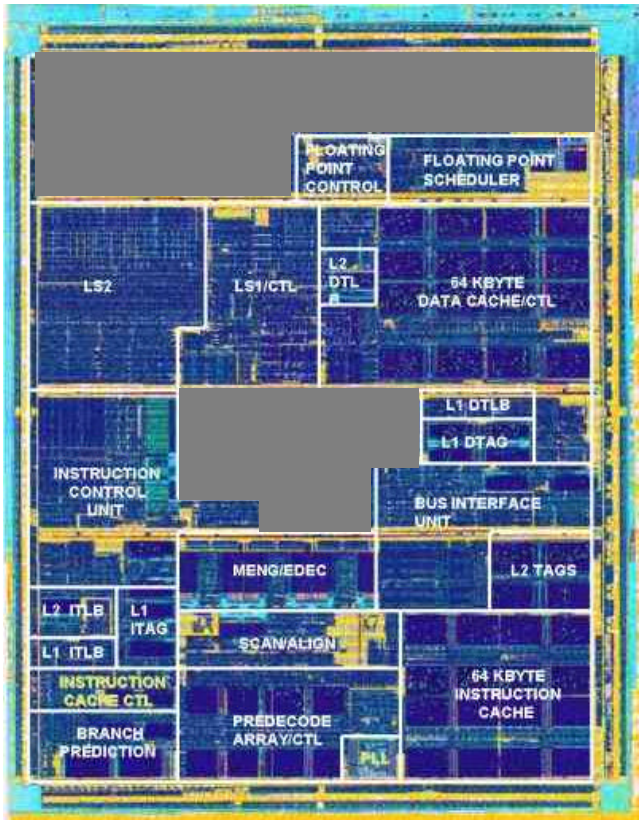
- Transistor counts have increased 1000x in 20 years
  - 1978: 29K in Intel 8086
  - 1999: 23M in NVIDIA GeForce 256



8086

- Suppose someone gave you 100M transistors and you had never seen a load-store architecture. What would you build?
- There is a need for designs scalable to 1G transistors and beyond

Motivation - Area Efficiency



K7 Die Photo

- In modern processors < 25% of chip area devoted to useful work (red areas on die shot)
- > 75% devoted to making the 25% faster
- Even the 25% is bloated due to
  - large instruction sets
  - complex superscalar design
- ...not necessarily a bad thing

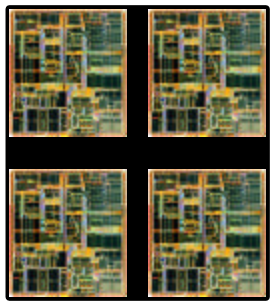
Motivation - Scalability

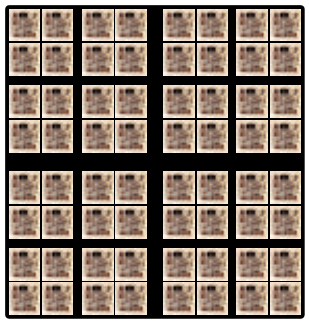
Today

- 1-4 processors per die
- Use available area to make processor fast
  - complicated designs

Tomorrow

- N processors per die
- Use available area for lots of processors
  - simple designs





Motivation - RAM Integration

- Logic and fast DRAM on a single die provides:
  - lower latency access to memory (~10x)
  - much higher bandwidth (~10x)
- What architectural features are enabled by this technology?
  - SRAM-tagged DRAM
  - Transactional memory
  - Forwarding pointers

Research Goals

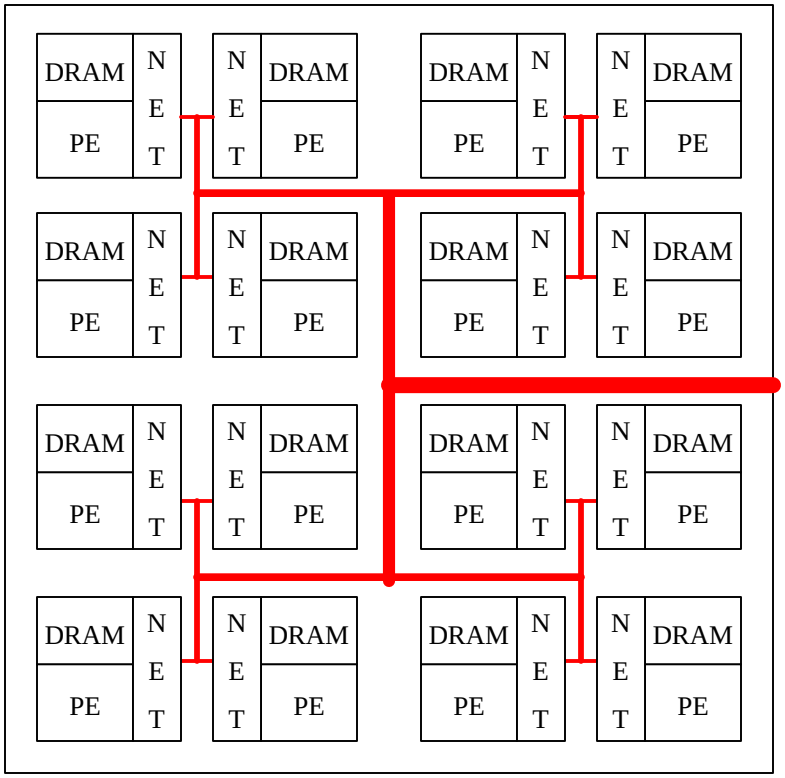
- Highly programmable parallel system
  - language, compiler, OS, architecture
- Manage huge data sets
- Area efficient architecture
- Design efficient architecture
- Integration of processor and memory
- Efficient Dynamic Fault Tolerance
  - A 1M node system is a great cosmic ray detector

Architectural Goals

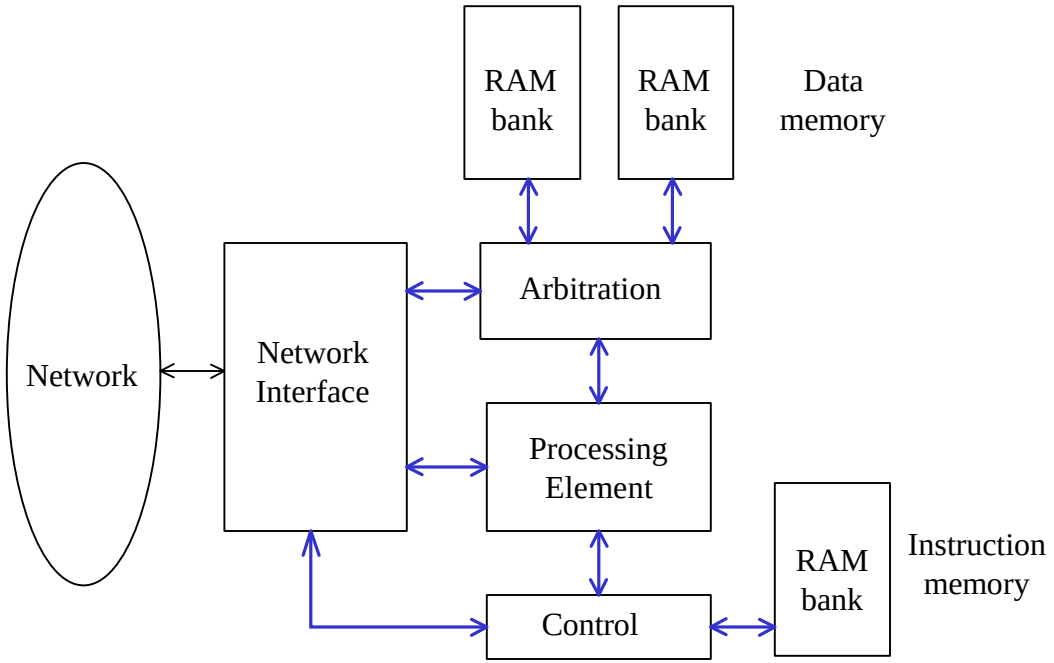
- Fast general purpose processor
  - ~1 GFLOP/processing node
- High performance network
  - high bandwidth (8GB/s at each node)
  - low latency (~40ns across a 1000 node system)
- Compilability
- Scalability
  - >> 1,000,000 nodes

Chip-Level View

- Array of processor-memory nodes connected by on-chip network
- Each node consists of processor, memory, and a network interface
- No caches



Processor-Memory Node



Processor Memory Integration

Once upon a time...

- Pozzo is designing hardware. He is assisted by the omnipotent **God O**f **T**echnology (Godot)

Pozzo: "I want memory!!"
Godot: "Here's your memory. Give it an address and it will give you data"

- Life is good

PMI con't

Pozzo: "I want virtual memory!!"
Godot: "Here's a page table mechanism. The page tables will reside in memory. I'll give you a TLB so that memory access can still be fast."

Pozzo: "I want protection in a shared environment!!"
Godot: "I've given every process a separate set of page tables. You'll have to clear the TLB when you do a context switch. Try not to do any context switches."

PMI con't

Pozzo: "I want data protection on a sub-page granularity!!"
Godot: "Here's a segmentation mechanism. Each segment has a descriptor with base, bounds and permissions. I've added all this to the TLBs. Try not to use too many segments"

Pozzo: "I want all this in a distributed memory system!!"
Godot: "Alright, just make sure you keep the TLBs globally consistent"

PMI con't

Pozzo: "I want data migration and forwarding pointers!! I want to stripe contiguous data across my nodes!! I want an object-based protection scheme!! I want to share data between processes on a sub-page granularity!! I want my system to be scalable!!"

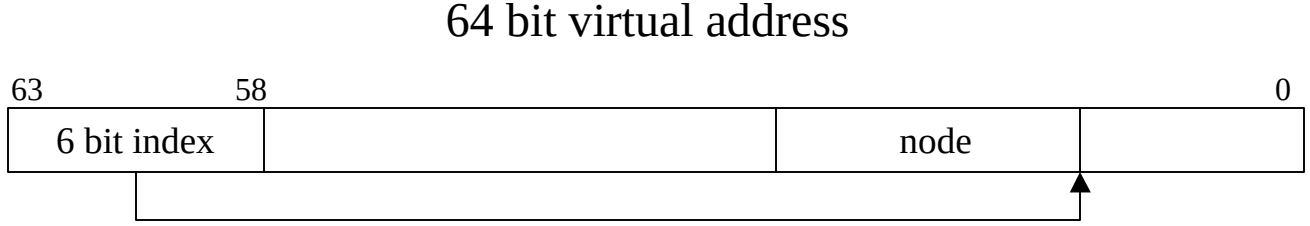
Godot: "I'm taking away your memory"

- Problem: Complexity layered on archaic model
- How can we develop a better model?

Capabilities

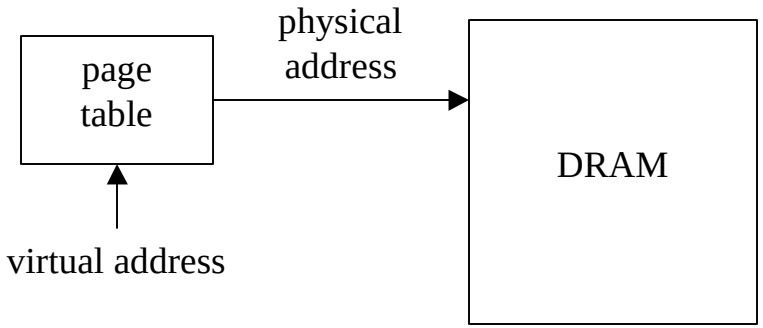
- Replace 64 bit pointers with 128 bit unforgeable capabilities containing:
  - 64 bit virtual address
  - base and bounds information
  - permission bits
- Allows global virtual address space
- Allows object-based protection
- Data sharing is easy

Multistriped Addressing



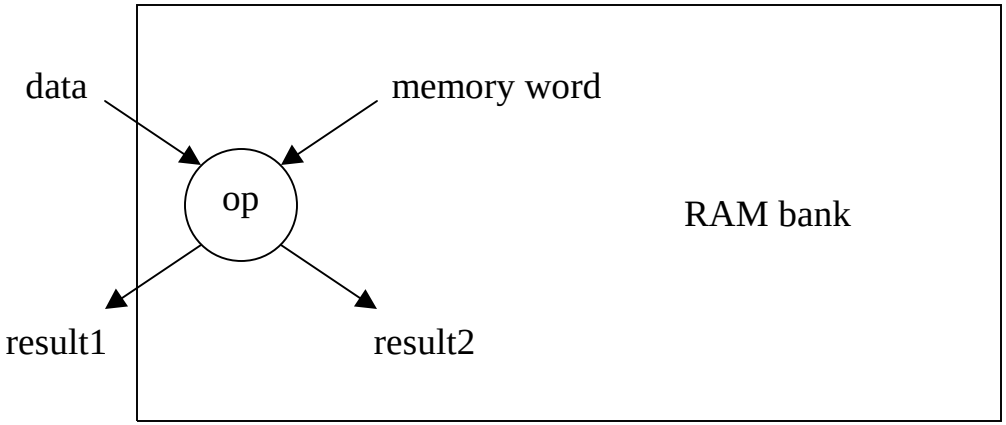
- Fixed virtual address → physical node maping
  - Eliminates the need for global tables
- Embed the node index within the virtual address at a controllable offset
  - Flexible data striping with any power of 2 granularity

Hardware Page Tables



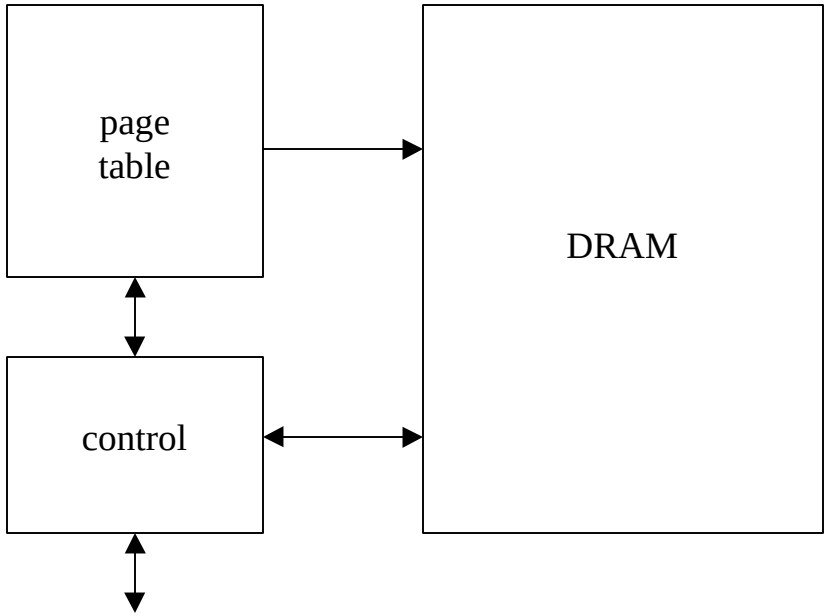
- Fully associative SRAM memory containing one entry per physical page
- Eliminates TLBs

Atomic Memory Operations



- Simple operation: boolean, addition, or copy
- Return nothing, result or previous memory word
- Store data or result

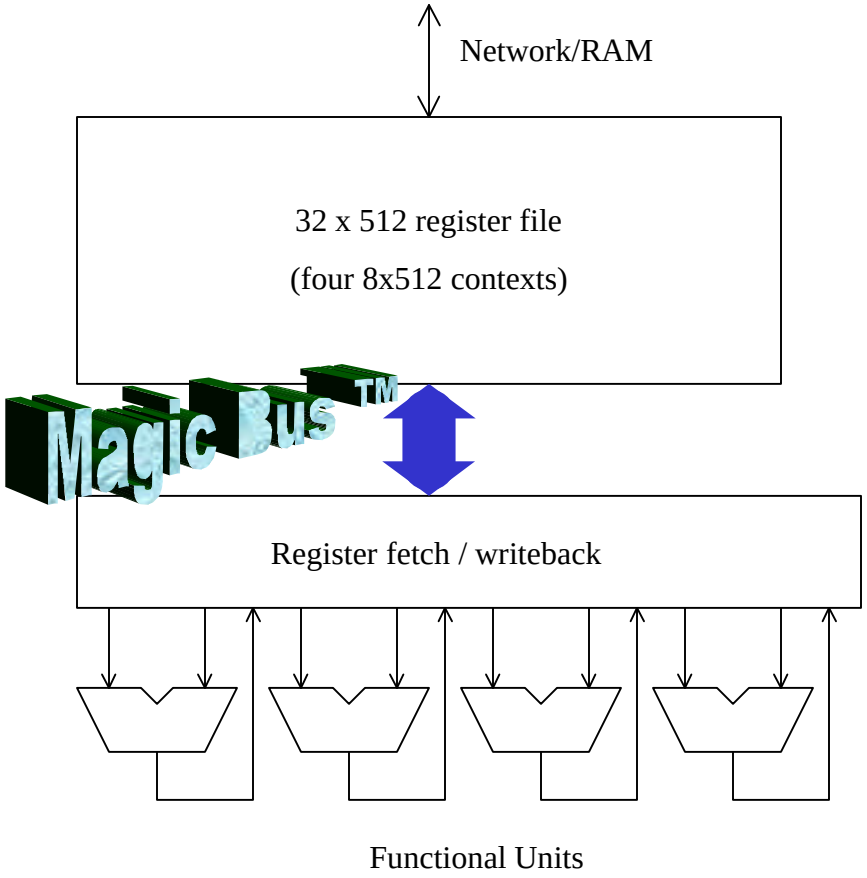
RAM Bank Overview



Processing element

- 128 bit multi-context VLIW processor
- ~32 registers per context
  - or 64x64b or 128x32b
- Four functional units
  - Multiplier/Adder (integer/floating point)
  - Adder (integer/floating point)
  - Boolean/Shift
  - Divide Square Root

Processing Element



Design Issue

- How to connect registers to functional units?
- Full connectivity
  - Easy for the compiler
  - Hard for the hardware designer
    - Lots of read/write ports!!
    - Lots of wires!!
- Partial connectivity
  - Can get away with MUCH less hardware
  - More challenging for the compiler

Simple Silicon

- No dynamic superscalar issue
- No out of order issue
- No register renaming
- No branch prediction
- No speculative execution
- No remote-data caching

Research Opportunities

- Processor evaluation
  - hand coded sample apps
  - performance at various design points
- Language design/evaluation
  - expresses parallelism and communication
  - programmability vs. compilability

Research Opportunities 2

- Compiler Design
  - C++ compiler
  - PSCHEME compiler
  - VLIW scheduling
- Operating System Design
  - Thread management, page management, memory management, etc. etc. etc.

Research Opportunities 3

- Alternative Execution Domains
  - secure computing
  - efficient dynamic typing
  - transactional/speculative computing
  - dataflow
- Hardware Design
  - Arithmetic circuit design
  - Verilog coding