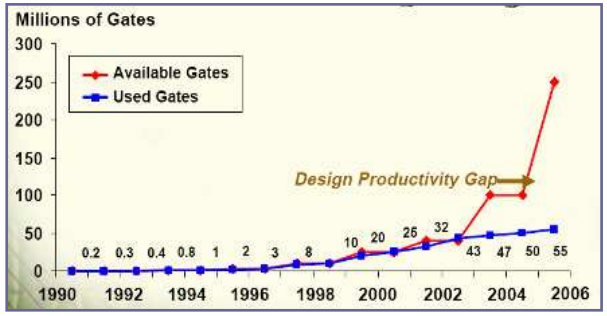
TeraOPS Hardware: A New Massively-Parallel MIMD Computing Fabric IC

Anthony Mark Jones
Mike Butts

Embedded Computing Problem

Traditional architectures are reaching limits in performance, scalability and ease of development

- Single CPUs and DSPs are reaching limits of extending performance
- Ordinary multi-core processors won't scale very far
- ASICs and high-end FPGAs are getting harder and more costly to develop

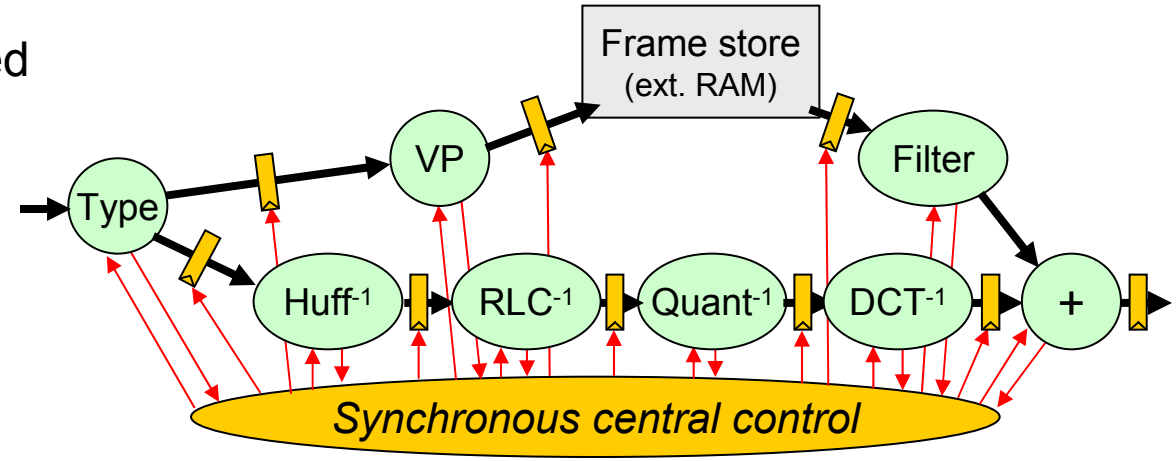


Gary Smith, Gartner Dataquest, DAC 2003

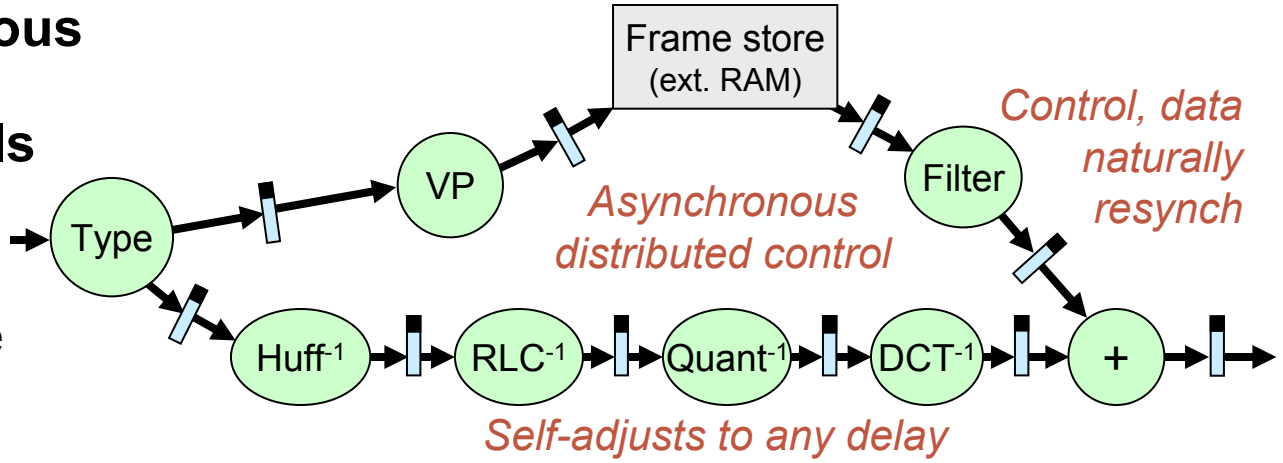
Embedded Synchronous Problem

- **Ordinary globally synchronous system**
  - Central controller required
  - Synchronization is up to the developer
  - Local changes break global synchronization
  - Difficult to validate
  - Not scalable

MPEG2 Decoder example

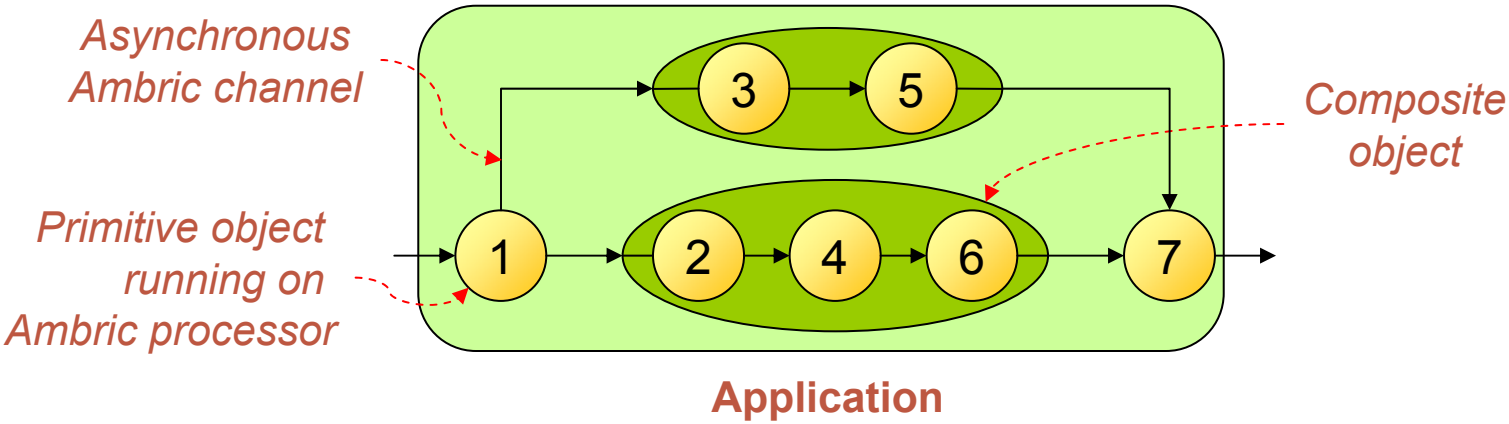


- **Globally asynchronous system of Ambric objects and channels**
  - Distributed control
  - Self-synchronizing
  - Local changes have only local effects
  - Scalable

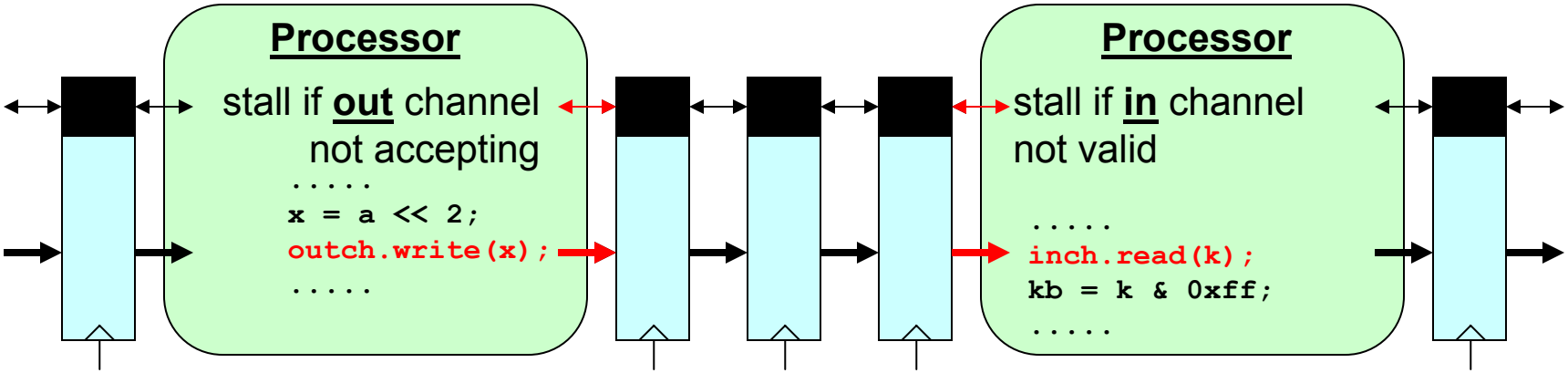


Ambric's Structural Object Programming Model

- Objects are software programs running concurrently on an asynchronous array of Ambric processors and memories
- Objects exchange data and control through a structure of self-synchronizing asynchronous Ambric channels
- Objects are mixed and matched hierarchically to create new objects, snapped together through a simple common interface
- Easier software development, high performance and scalability



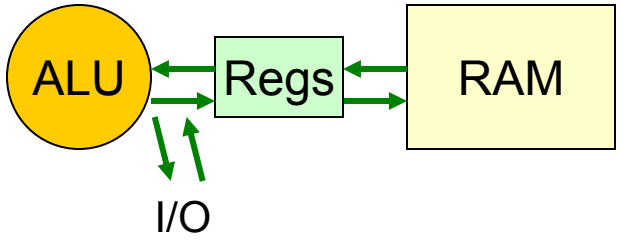
Ambric Registers and Channels



- **Ambric register**
  - Latency, speed, area, power of an ordinary register
  - FIFO buffering, localized forward and back-pressure
- **Chains of Ambric registers form Ambric channels**
  - Fully encapsulated, fully scalable for control and data between objects
- **Ambric processors are interconnected by Ambric channels**
  - Ambric registers permit inputs and outputs to run at different clock rates

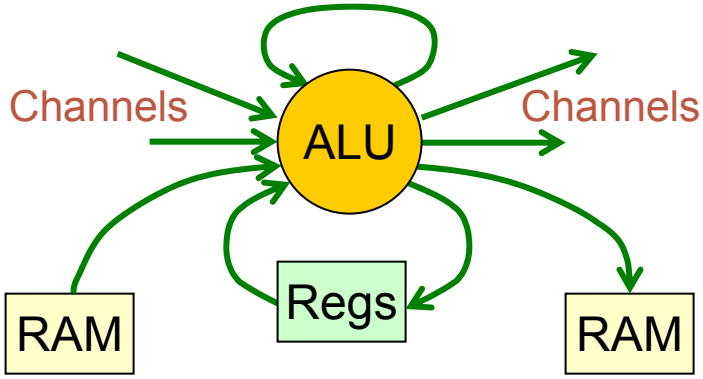
Traditional vs. Ambric Processors

- **Traditional processor architecture**
  - Primary: register-memory hierarchy
  - Secondary: communication



- **Ambric processor architecture**
  - Primary: communicate through channels
  - All data goes through channels
    - Memory
    - Registers
    - Inter-processor streams
    - Instruction streams

    to reduce local storage
  - Channels synchronize all events
  - ELIO: Execute/Loop/Input/Output every cycle



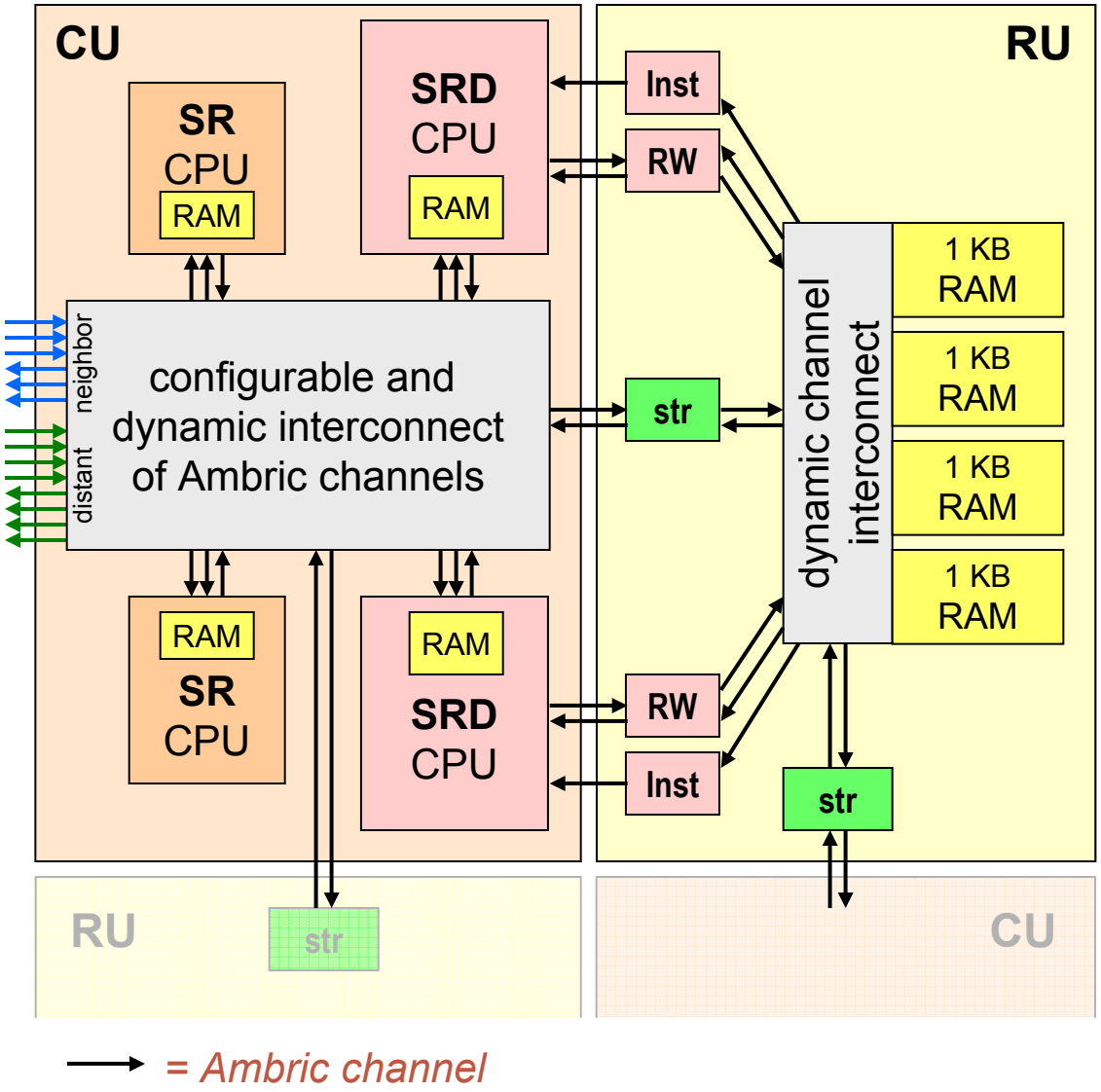
Compute Unit, RAM Unit

Compute Unit (CU)

- SRD 32-bit CPU
  - *Streaming RISC with DSP extensions*
  - *3 ALUs: 32b, 2x16b, 4x8b*
  - *256 word local RAM*
- SR 32-bit CPU
  - *Streaming RISC*
  - *1 ALU : 32b, 2x16b*
  - *64 word local RAM*
- 32-bit Ambric channel interconnect
  - *Processor-Processor*
  - *CU-Neighbor*
  - *CU-Distant*

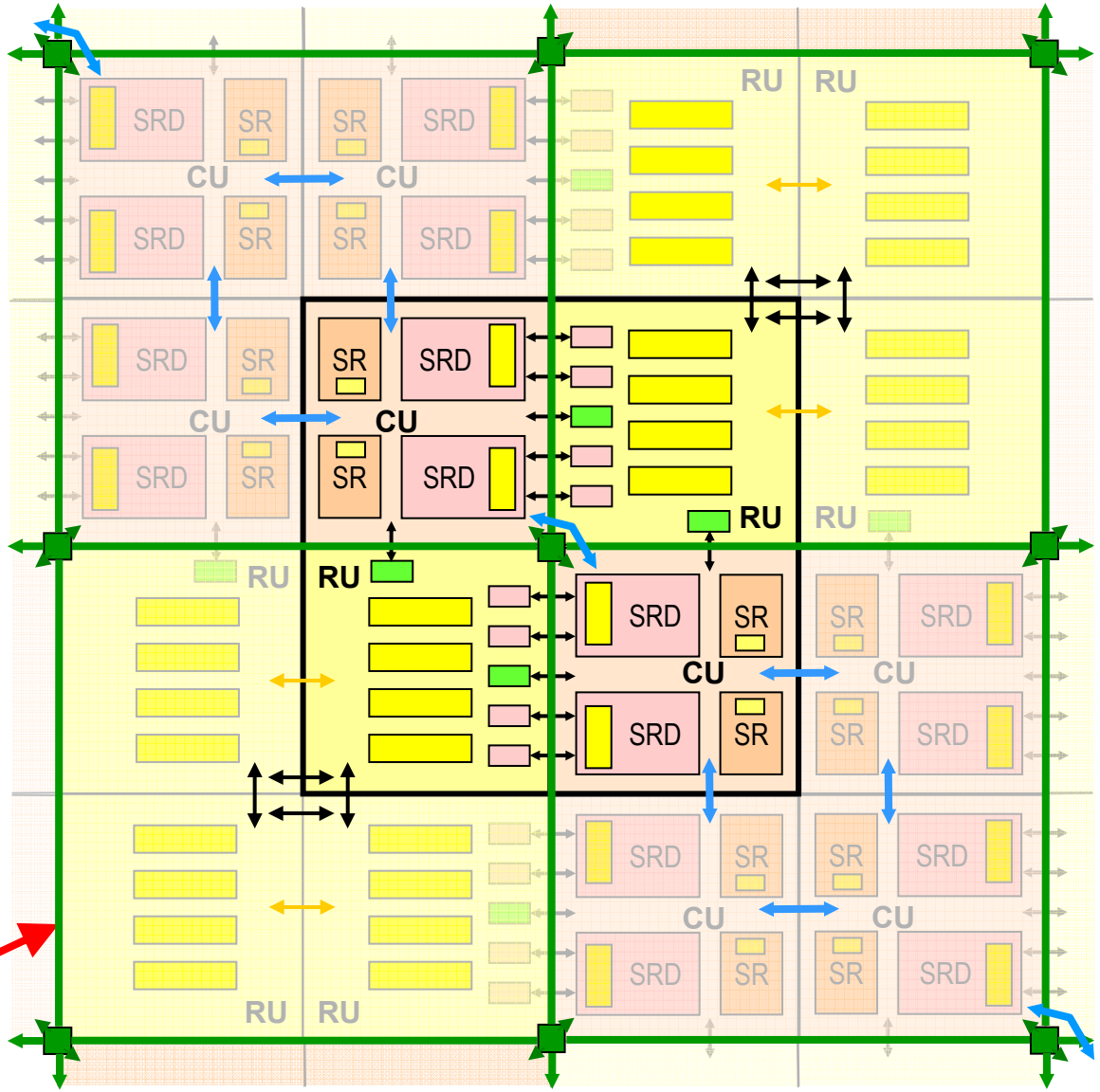
RAM Unit (RU)

- Four 1KB RAM banks
- RU engines
  - *RAMs, FIFOs, etc.*



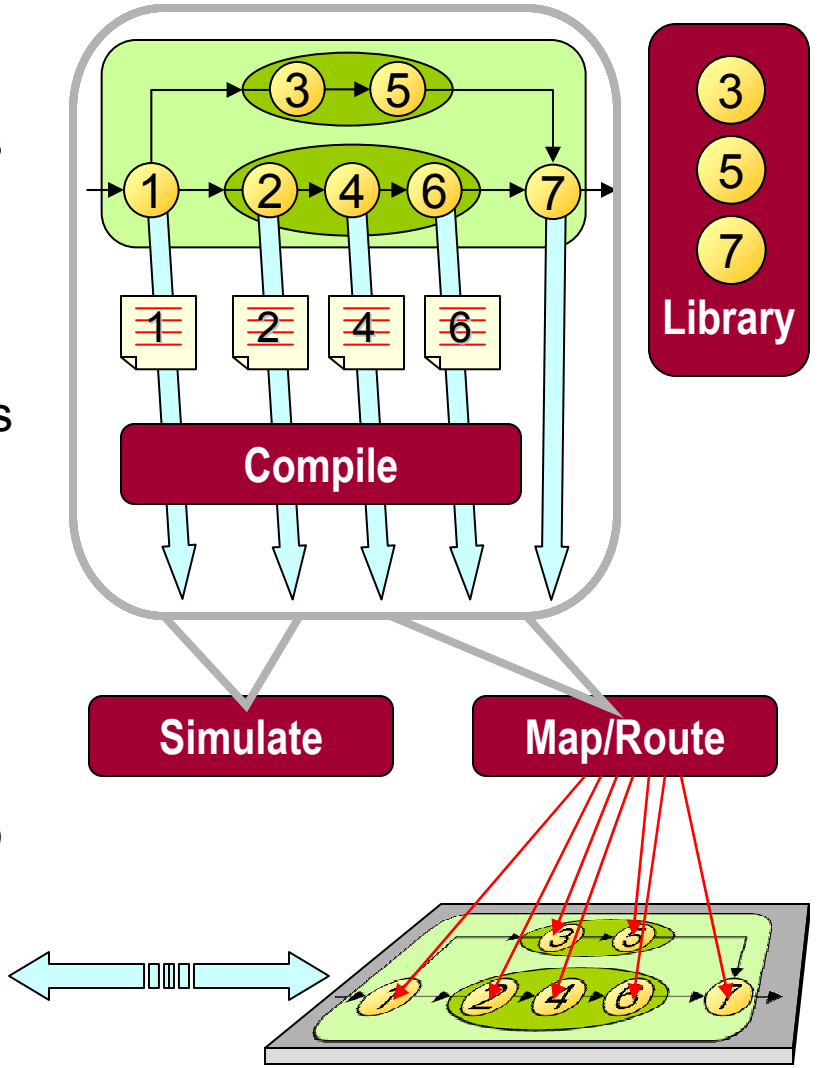
Brics and Interconnect

- The bric is the physical building-block
  - Two CU-RU pairs
  - 8 CPUs
  - 13KB RAM
- Brics connect by abutment to form a core array
  - CU quads, RU quads
- Neighbor CU channels
- Distant CU channels:
  - bric-length hops
  - configurable switches
- No wires longer than a bric



Programming Model and Tools

- **Structure**
  - Conceive your application as a structure of objects and the messages they exchange when running
  - Divide-and-conquer using hierarchy
- **Reuse**
  - Validated, encapsulated library objects
- **Code and test**
  - Write your application-specific objects and compile
  - Verify with functional simulation
- **Realize**
  - Run mapper-router and configure chip
- **Run and Visualize**
  - Observe and control objects and messages using dedicated debug HW

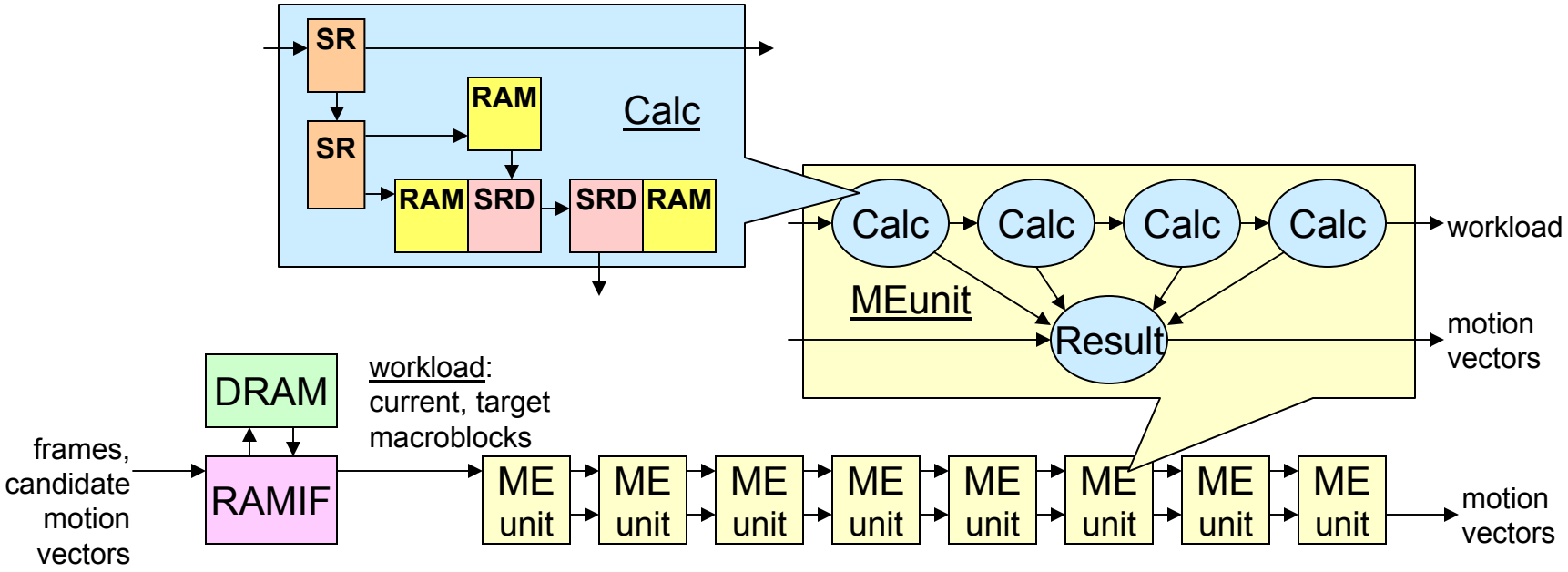


Performance Metrics

- **Prototype chip @ 45 brics:**
  - 1.08 trillion operations per second (24 BOPS per bric)
  - 425 Gbps interconnect bi-section bandwidth
  - 26 Gbps DRAM + 16 Gbps high-speed serial + 13 Gbps parallel

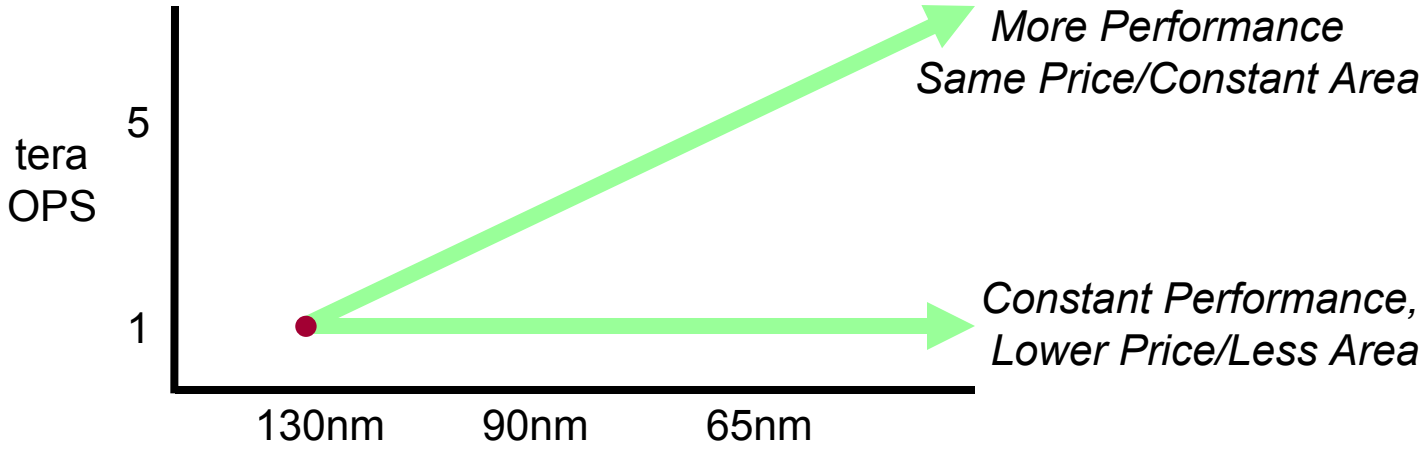
	Ambric 45-bric Prototype	Ambric 70-bric Example	TI C641x DSP	Xilinx Virtex-4 LX100 - LX200
Process	130nm	90nm	90nm	90nm
MHz	333 MHz	Est. 450 MHz	1,000 MHz	500 MHz
Published DSP Benchmarks	10-25X throughput, 1/3 the code	Est. 20-50X throughput, 1/3 the code	1X	n/a
Multiply-Accum./Sec. (16x16 – 32-bit)	60 GMACS	Est. 125 GMACS	4 GMACS	48 GMACS

Application Example: Motion Estimation



- Perform SAD calculation for 16x16 macroblocks, choose best results
  - Exhaustively over +/-16 pixel range, centered on *any* candidate vector
  - For 720p (1280x720) @ 60 frames/sec, two reference frames
- Actual performance using 89% brics @ 300 MHz: 0.46 teraOPS
  - 53% of maximum teraOPS available

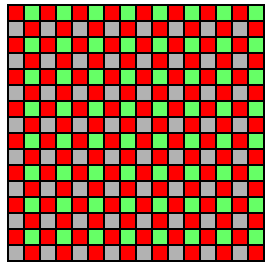
Intrinsically scalable to 65nm and beyond



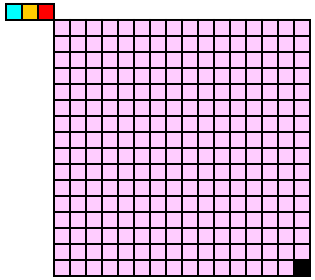
- Hierarchical Object-Based Modularity for Development Cost
  - Massive design reuse requires strict encapsulation, simple identification of dependencies and local synchronization
- Communication-Centric Design for Timing Scalability
  - Globally asynchronous, locally synchronous (GALS)
- Massive Parallelism for Power Scalability
  - MIMD architecture: power scales linearly with performance

Other Massively-Parallel Architectures

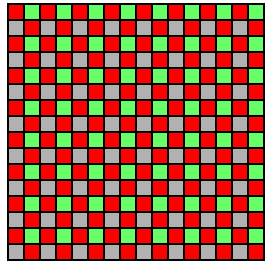
- **Ambric architecture is a member of an emerging class: *Reconfigurable Processing Array* (RPA)**
  - Hundreds of processing elements such as CPUs, ALUs, memories…
  - Rich, word-wide, reconfigurable interconnect fabric
  - **SIMD control** or **MIMD control**



- **MIMD is more general than SIMD**
  - MIMD is effective on irregular complex apps (H.264)
  - Efficient on other data structures than just vectors
  - Processors stay busy on different size data sets
  - Processors do not have to branch in lock-step

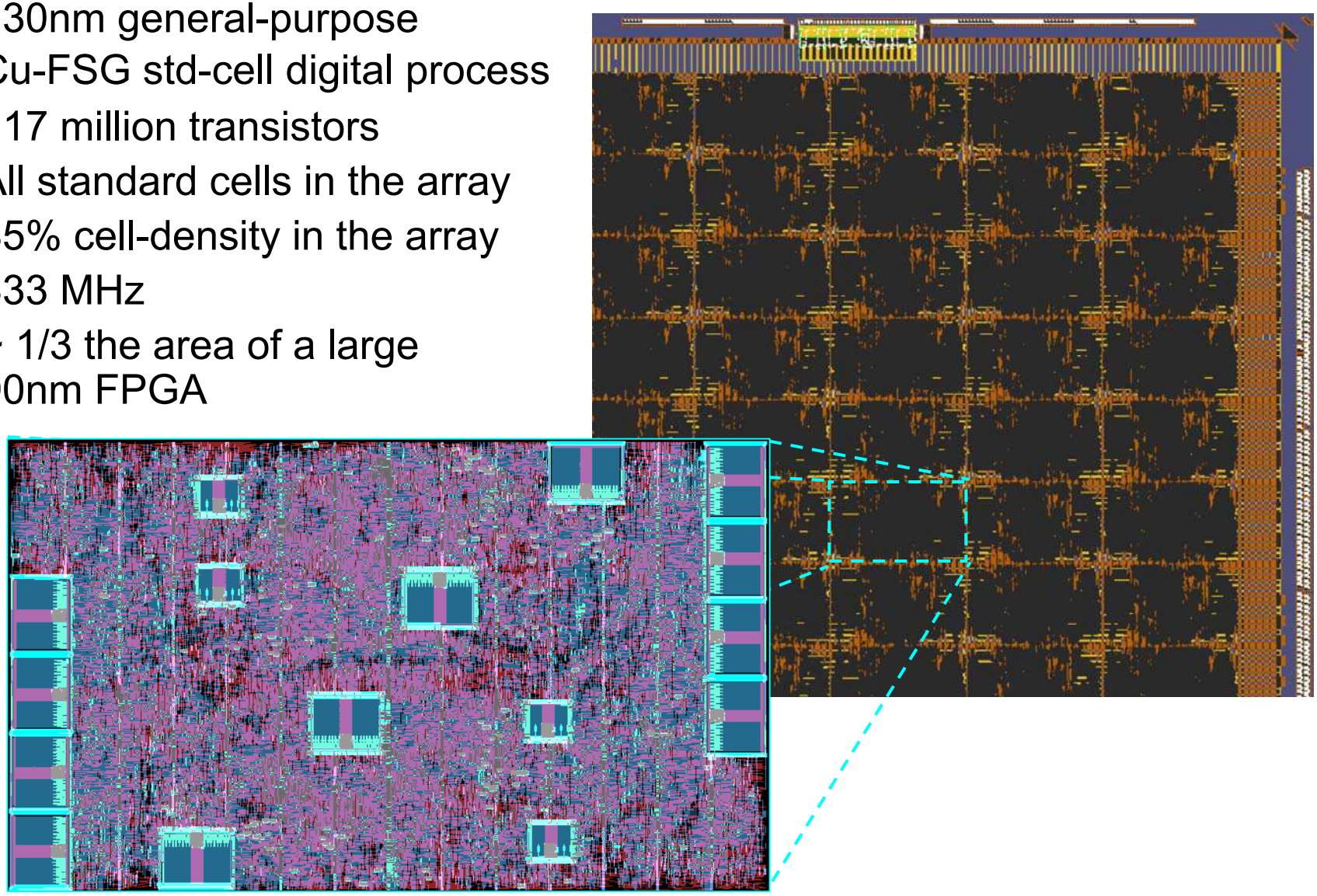


- **Ambric's MIMD RPA is practical**
  - Interconnect is dynamically self-scheduling
  - Based on an asynchronous parallel model of computation
  - Standard high-level language (strict subset) or assembly
  - Globally asynchronous: efficient and scalable



‘Kestrel’ Prototype IC

- 130nm general-purpose Cu-FSG std-cell digital process
- 117 million transistors
- All standard cells in the array
- 85% cell-density in the array
- 333 MHz
- ~ 1/3 the area of a large 90nm FPGA



Summary

- Ambric has solved many of the architectural and programming challenges of massively-parallel embedded computing
- Ambric's chip and tools realize a Structural Object Programming Model for ease of development
- Ambric chip and tools deliver 10X+ performance over traditional alternatives for high-performance embedded computing
- Ambric's architecture economically scales with Moore's Law

Patents pending