
Streaming Supercomputer Strawman

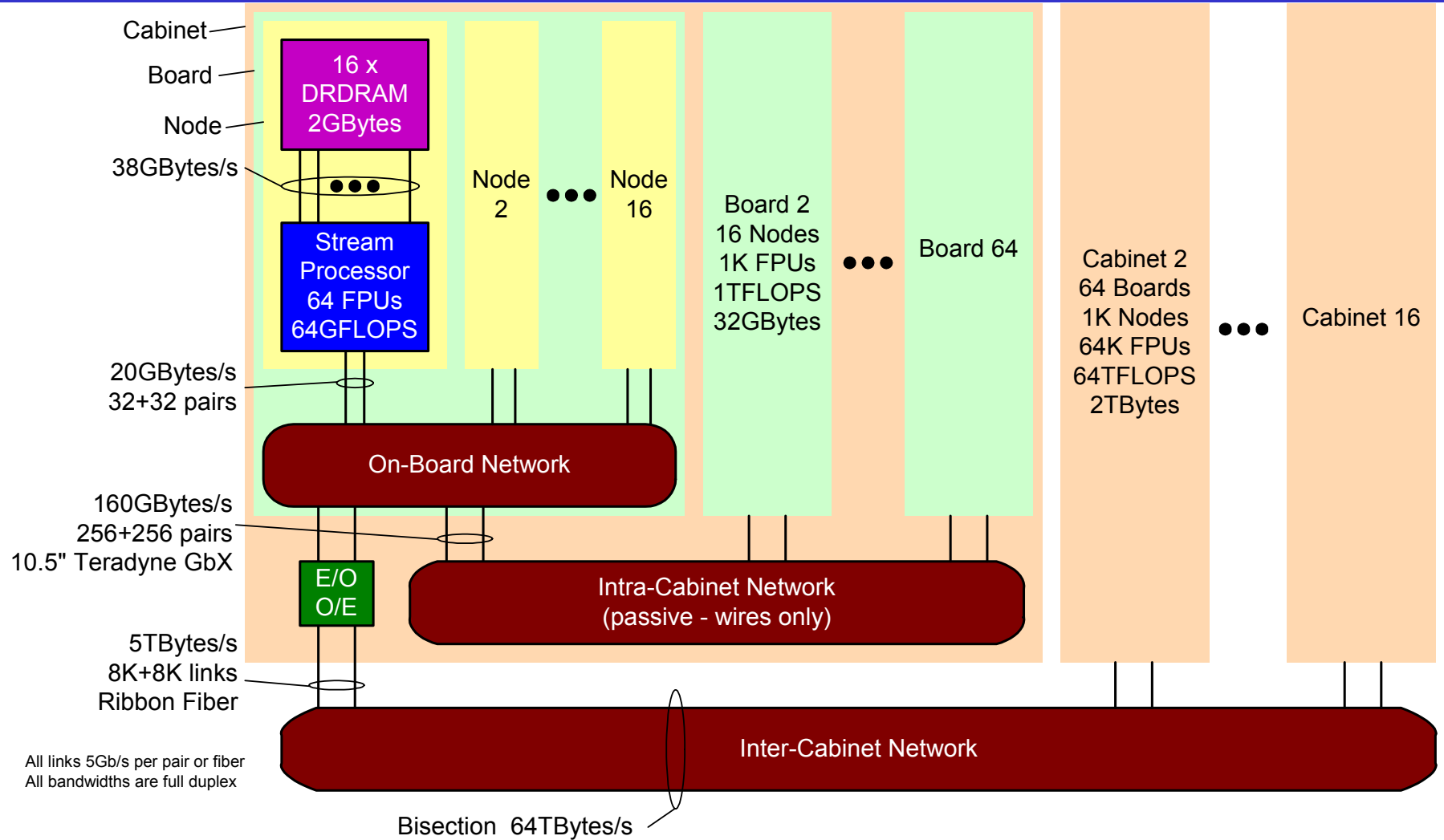
Bill Dally, Jung-Ho Ahn, Mattan Erez,
Ujval Kapasi, Tim Knight, Ben Serebrin

April 15, 2002

Outline

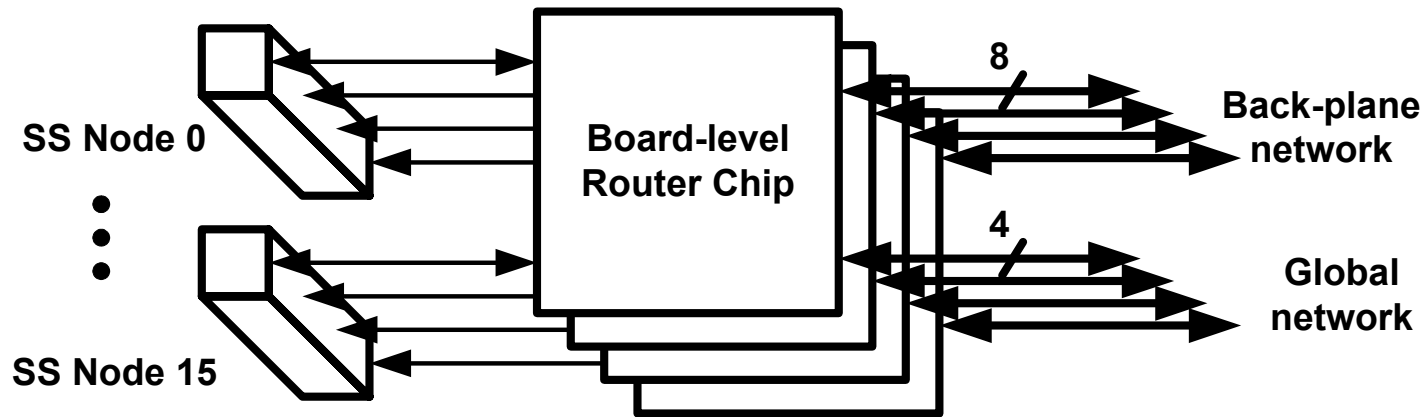
- Overview
- Operation
- Stream-ISA
- Kernel-ISA
- Micro-architecture
- Next steps

System Overview



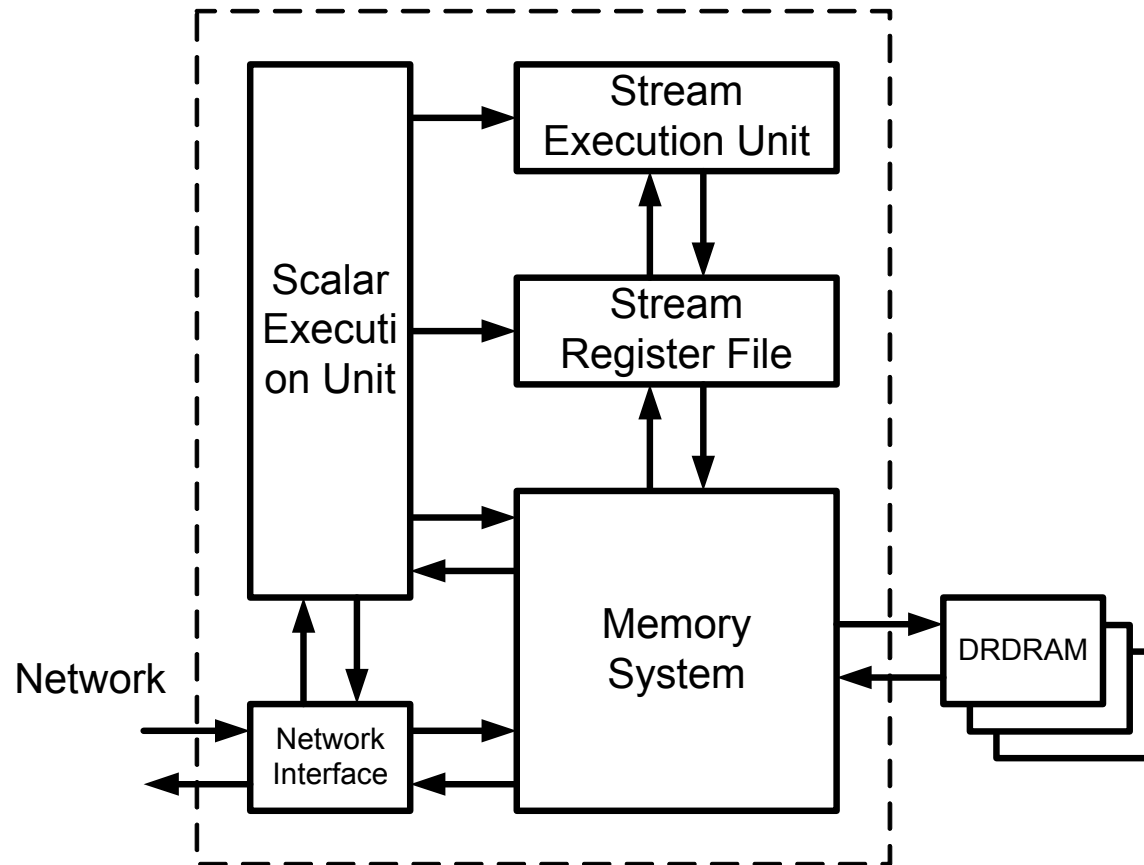
Roundtrip memory access latency $\sim 500\text{ns} = 500$ processor cycles

Board Overview

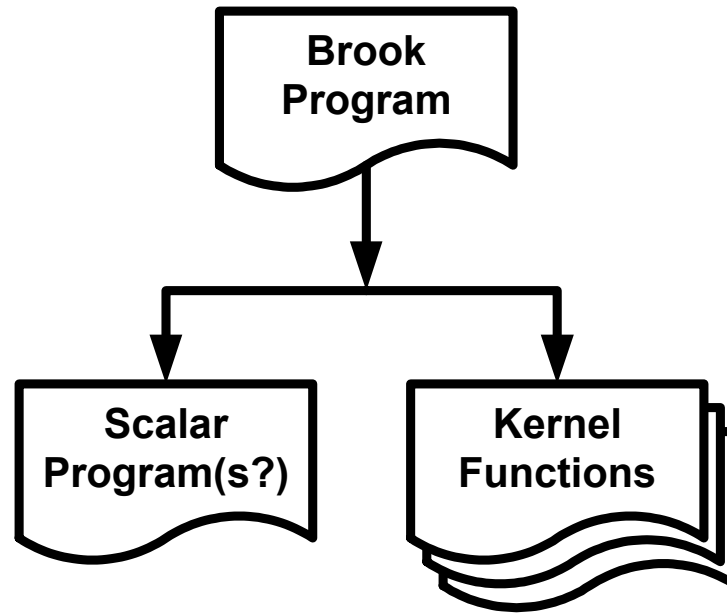


- Chips
 - 16 SS nodes
 - 4 router chips – provide 4 independent routing planes on each board
- Ports
 - 32 to back-plane network
 - 16 to global network

Node Overview



Node Operation [1]

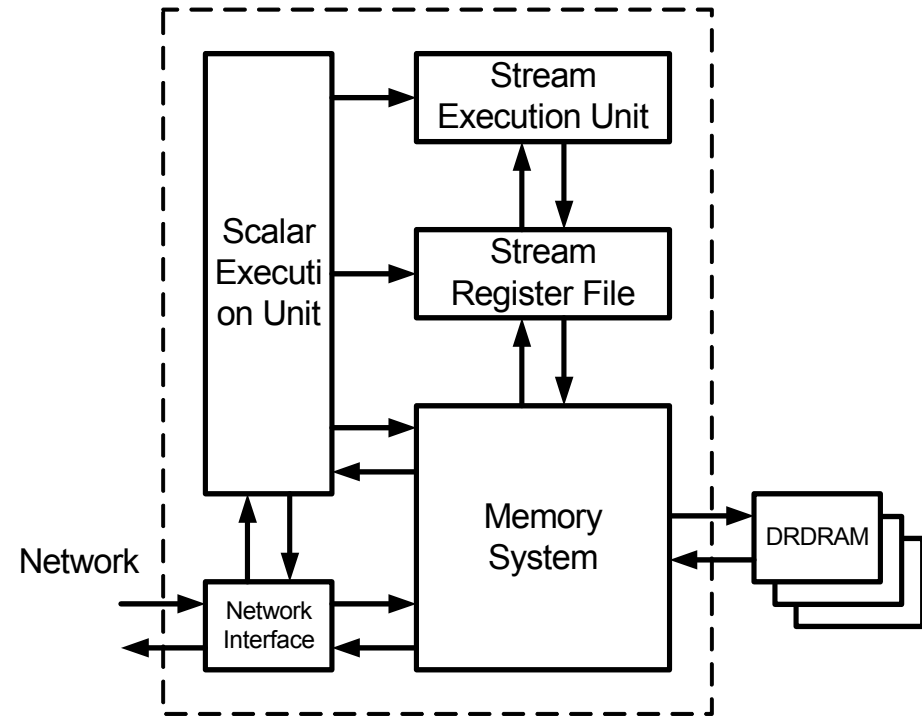


- MIPS assembly
- COP2 instructions encode stream instructions
- VLIW microcode
- Called by instructions in the scalar program

Node Operation [2]

Example Execution

- Transfer data from DRDRAM and network into SRF
- Execute Kernel $k1$
- Execute Kernel $k2$
- Synchronize (across nodes)
- Store results to DRDRAM and network
- Synchronize



Stream-ISA Visible State

State which is used by a scalar (MIPS) program:

- MIPS registers
 - Standard processor registers
 - Coprocessor 2 interface registers – the MIPS program's interface to the SSS
- SRF
- Global memory: all memory on all nodes can be addressed by any node
- Control registers:
 - *Segment registers*: implement virtual memory
 - *Stream descriptor registers & memory descriptor registers*: hold parameters such as length, record size, etc.
- Stream cache

Stream Instruction Set

- MIPS instructions
 - Standard processor instructions
 - COP2 instruction used to issue stream instructions
- Stream instructions
 - Write control registers
 - Stream load, store
 - Stream cache prefetch, invalidate
 - Kernel load, execute
- More on...
 - Messaging instructions
 - Global Synchronization
 - Exceptions/Interrupts
- Open Issues
 - COP2 interface
 - Critical sections for sending stream instructions

Memory Model

- Address space shared by scalar/stream units
- No data-duplication of global data in local DRAM
- Virtual memory implemented via programmable segment registers
 - Virtual address: (*SegNum*, *SegOffset*)
 - Physical address: (*NodeNum*, *LocalOffset*)
- Segments can be interleaved across nodes in user-specified amounts.
- Read-mostly stream cache with gang invalidation
 - MIPS instructions specify which stream elements to cache
- No HW paging support
- Open Issues
 - Cache write policy
 - Coherence

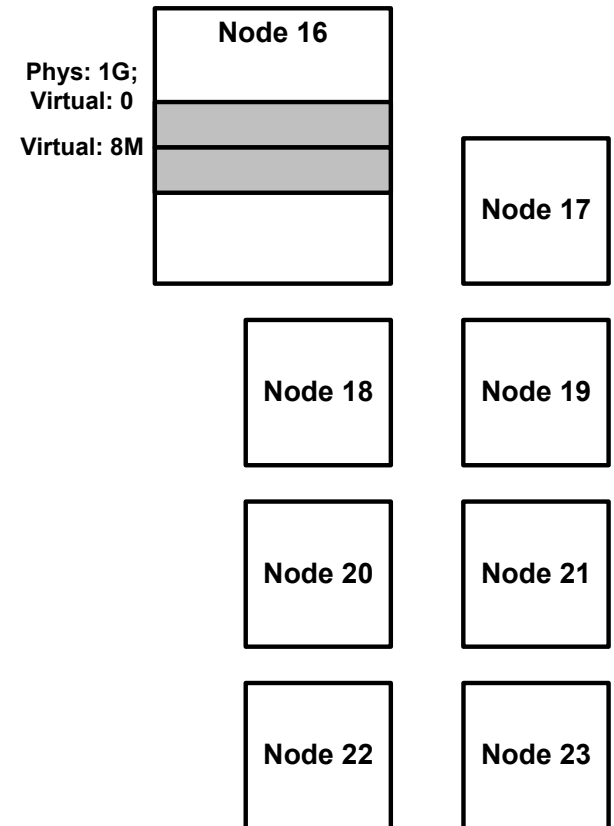
Segment Register:

NodeBase = 16

Size = 256MB

Base = 1GB

NodeBits = 20-22



Global Synchronization

- Minimal global synchronization and communication mechanisms
 - Barrier
 - Remote update - Fetch-and-add/Compare-and-swap
- Implement all other synchronization primitives in software
 - General messaging mechanism interrupts scalar processor
 - e.g., General-purpose synchronization
 - For example, end a parallel search as soon as one node finds a match
- Open Issues
 - Hardware acceleration
 - Locking: a table with 1K entries for locked locations

Exceptions/Interrupts

- Scalar processor handles all interrupts
 - e.g., Message received from remote node interrupts scalar processor
- Stream operations delay exceptions until the end of the operation
 - Examples: divide-by-0 in clusters, invalid address in memory system
 - Information about exception saved
 - Scalar processor must read this state to figure out nature of exception
- Open Issues
 - Best way to save info about stream operation exceptions and transfer this to scalar processor
 - Multithreading scalar processor

Kernel-ISA Visible State

State which is used by a kernel function:

- Per-cluster:
 - Local register files
 - Per ALU register files
 - Cluster condition code registers
 - Scratchpad: small indexable register file, used for:
 - lookup tables
 - complex data structures
 - register spills
- Per-node:
 - Microcontroller register file, used for:
 - Loop counters
 - Data to be broadcast to clusters
 - Passing parameters between the MIPS processor and the kernel functions
 - Microcode store contains the kernel VLIW instructions.
 - Microcontroller condition code registers, for looping and conditional streams
 - Microprogram counter

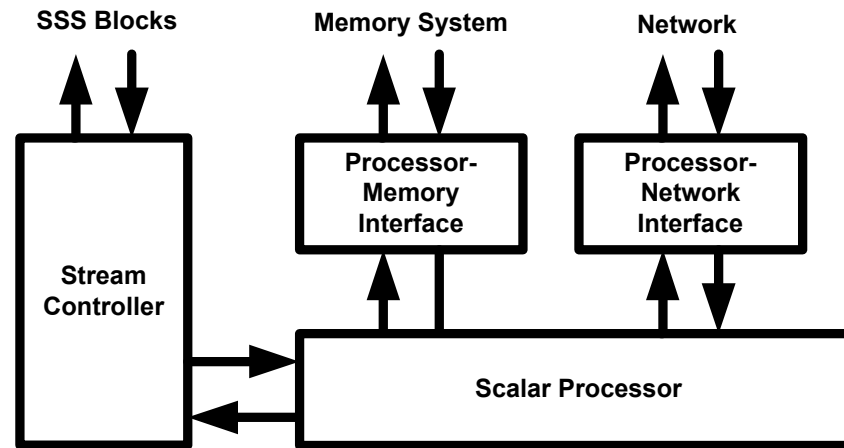
Kernel Instruction Set

- Kernel microprogram consists of VLIW instructions
- Each cluster in the node is sent the same control signals each cycle (i.e.) they execute in lockstep (SIMD)
- Kernel VLIW instructions control:
 - Microcontroller units and register files
 - Cluster arithmetic units and register files
 - Inter-cluster switch
 - Transfers of stream data between the clusters and the SRF
- More on...
 - Conditional Operations

Conditionals

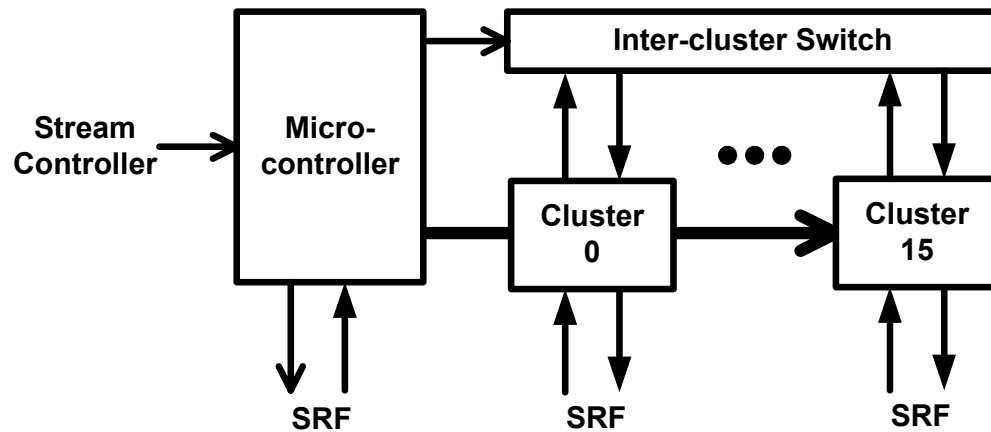
- Hardware select (analog to C “?:” operator)
- Scratchpad (using register indexing)
- Conditional Streams
- Open Issues
 - Assuming that Brook will retain ‘if-statements’
 - Need to automatically map code to conditional streams or predication
 - Requires a method to “split” a kernel

Scalar Execution Unit

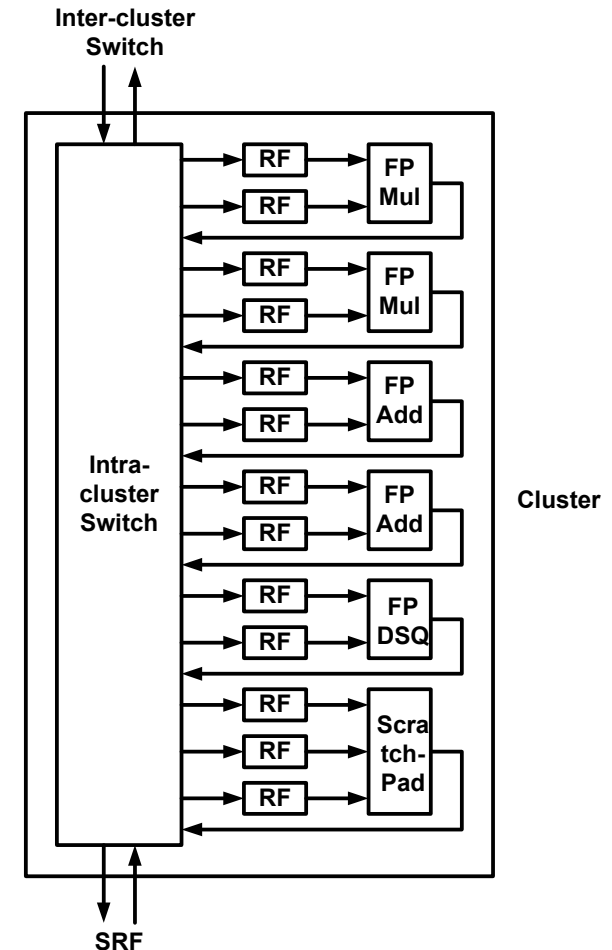


- Scalar processor is a MIPS (or Tensilica) core with data and instruction caches
- Scalar processor issues stream instructions to the stream controller
 - Stream Controller stores them in a scoreboard
 - Instructions issued when
 1. all required resources are available
 2. all inter-instruction dependencies have been satisfied.
- Open Issues
 - On-chip L2 cache for MIPS Core
 - Best method to encode dependencies between stream instructions

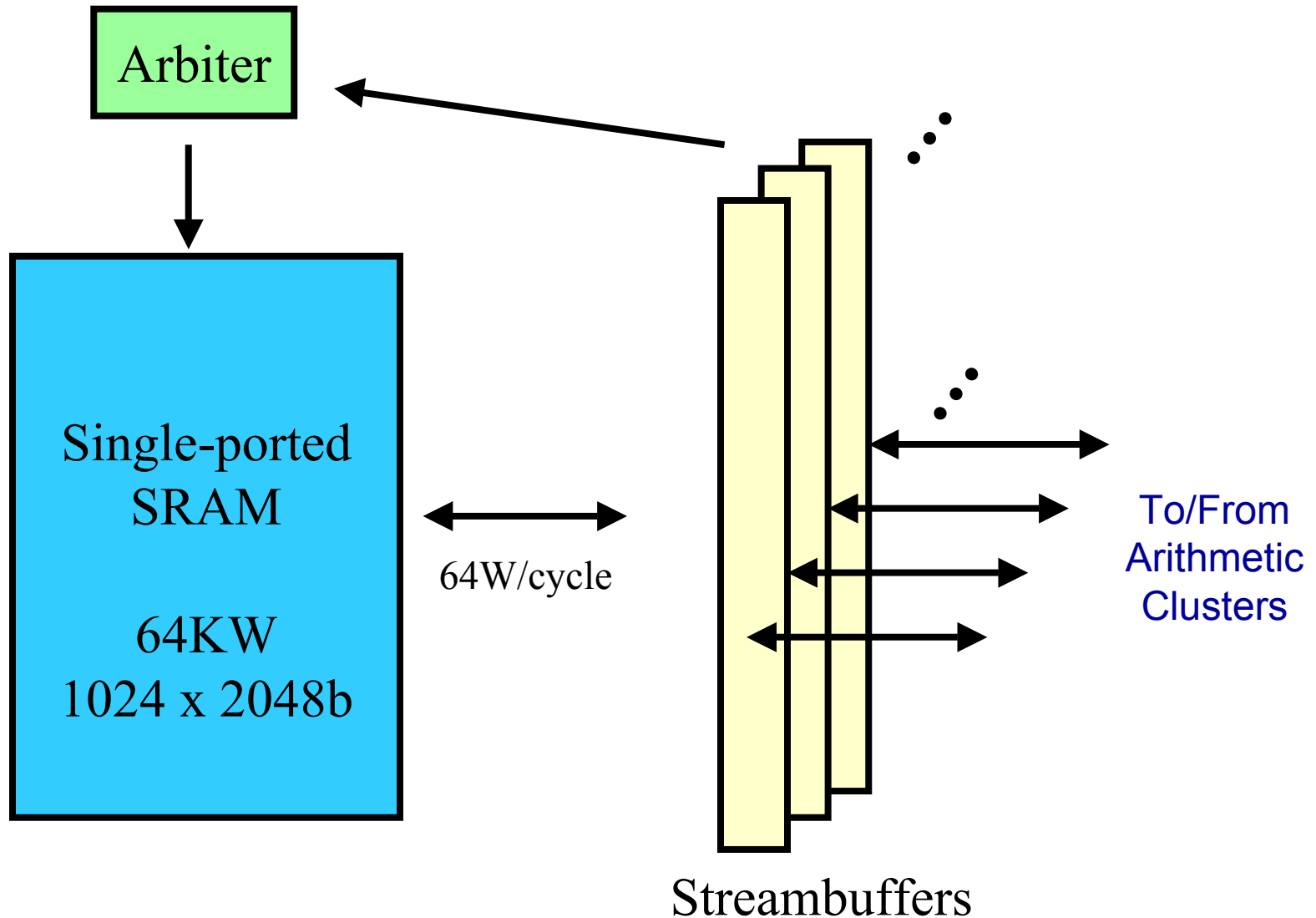
Stream Execution Unit



- Kernel instructions issued by microcontroller
 - SIMD clusters
 - VLIW control of ALUs within a cluster
- Open Issues
 - “Aspect Ratio”
 - Inter-cluster switch implementation
 - Local register file organization
 - FU Mix
 - Division / Square root support
 - Integer unit for logical ops

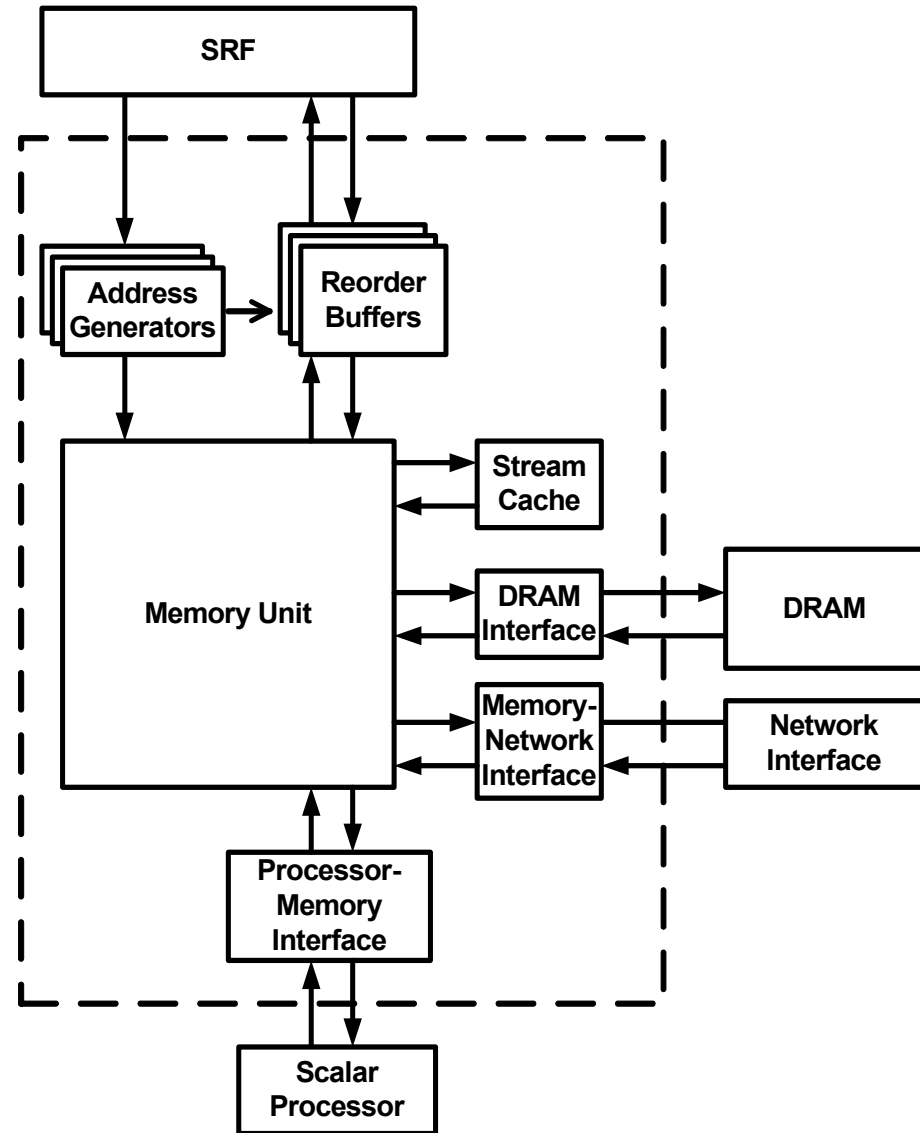


Stream Register File

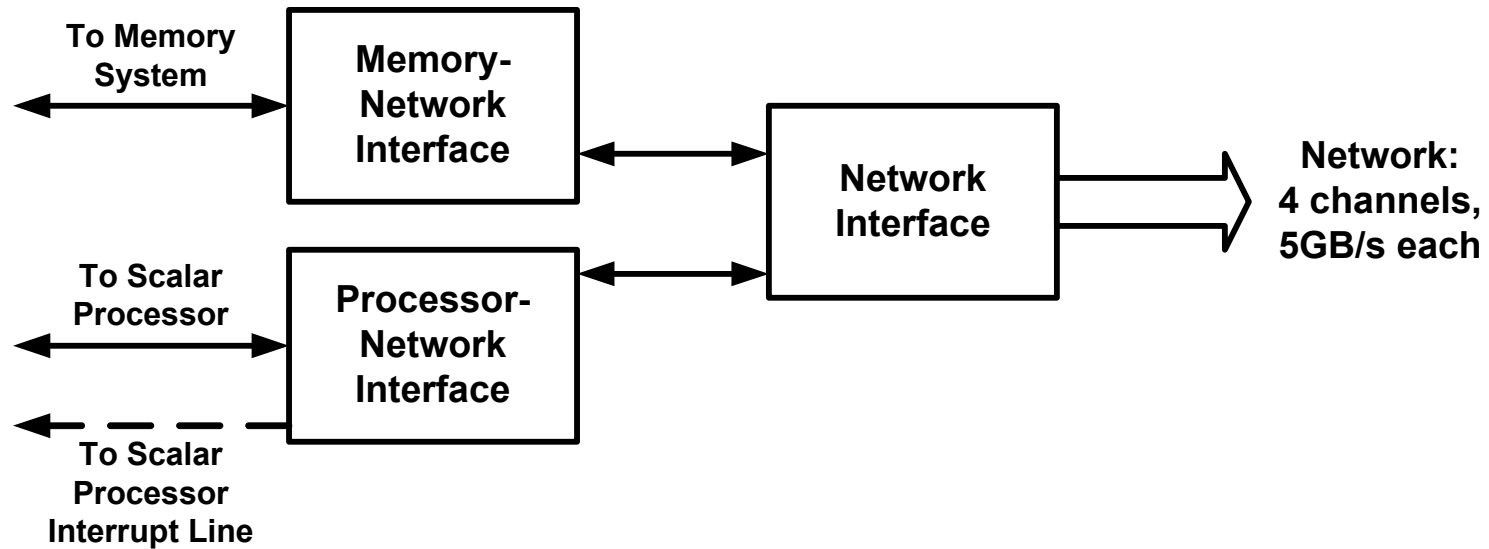


Memory System

- Memory Unit
 - Translates virtual addresses
 - Routes requests and replies
 - Frames new network message for each external word
- Requestors
 - Address generators
 - Scalar processor
 - Stream cache
 - Network
- Suppliers
 - Local DRAM
 - Stream cache
 - Network
- Open Issues
 - Multidimensional strides



Network Interface



- Flit-reservation flow-control
- Expect to be able to service messages faster than arrival rate
- Open Issues:
 - Need to flesh out the details of this module

Next Steps

- SVM will be used to study system-wide architecture
 - e.g. System bandwidths, synchronization mechanisms, etc.
- Cycle-accurate simulator (ssim)
 - Used for node architecture studies
 - Validate multi-node results
 - Current emphasis on getting single-node simulations to work
 - Feedback single-node results (i.e., kernel results) into SVM for quick but fairly accurate system-level results
- Global architecture studies
 - Feasibility, Global mechanisms, network topologies
- Node architecture studies
 - Aspect ratio, FU mix, inter-cluster switch, conditionals, SRF size/bandwidth
 - Most are gated on getting apps ported
 - We can start with Imagine apps for now
 - Area/Power estimates
- End-of-quarter project-wide goal is to have Brook apps running on SVM and ssim.
 - We should be able to conduct architectural experiments at that point