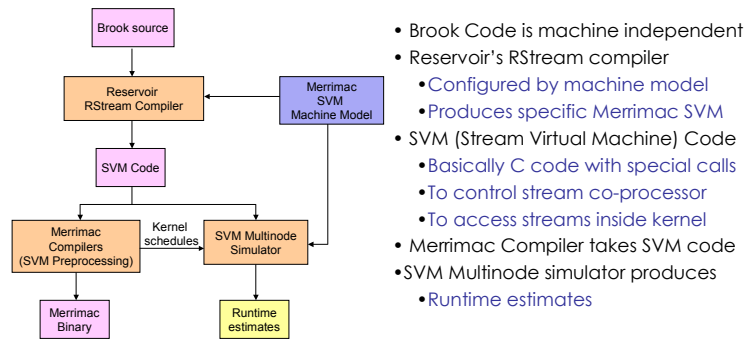


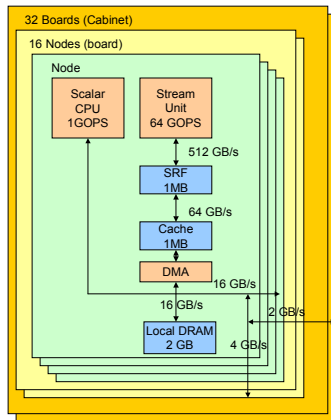
## Merrimac Compilation Model



- Brook Code is machine independent
- Reservoir's RStream compiler
  - Configured by machine model
  - Produces specific Merrimac SVM
- SVM (Stream Virtual Machine) Code
  - Basically C code with special calls
  - To control stream co-processor
  - To access streams inside kernel
- Merrimac Compiler takes SVM code
- SVM Multinode simulator produces
  - Runtime estimates

The Merrimac streaming supercomputer project requires tools to simulate correctness performance and evaluate implementation tradeoffs. The infrastructure of the Merrimac cycle accurate simulator is inherited from the Imagine project. In addition to Merrimac specific features, the Merrimac simulator also models the scalar code running on the RISC CPU. Cycle accurate simulation of a system with many nodes is not feasible so we calibrate a higher level simulator, the SVM simulator to run simulations with many nodes.

## Merrimac SVM Machine model



- Machine Model used to characterize targets of SVM code
- Describes both
  - Physical composition, connections
  - Performance characteristics
- Allows for Hierarchies (nice to model packaging hierarchy)
- Used by:
  - Rstream compiler to generate efficient code
  - SVM simulator to estimate run-times (bandwidth, kernel runtimes)

## SVM Simulator Runtime Estimates

- SVM simulator models run times of kernels
  - Use schedule from Merrimac kernel scheduler
  - Linear cost function model:
    - Fixed cost for kernel includes prologue and postscript
    - Per input stream element cost (inner loop)
  - Doesn't account for conditionals in kernels (possible extension)
- SVM simulator models transfer times of stream memory operations
  - Cache included in simulator, gets hit-rate on cached streams
  - Outstanding memory requests take a certain amount of time based on their locations
- SVM simulator doesn't currently account for run-time on scalar processor and memory traffic generated by it
  - Could use thread run-time to estimate it's run-time
- When we are waiting on concurrent (scalar, kernel, memory) operations, the longer one dictates the wait time.

## Stream Virtual Machine (SVM) Simulator

- SVM code is C with special API calls for control processor to control stream processor, for kernels to access streams
- With appropriate SVM library, SVM code can be compiled by a normal C compiler
- SVM simulator is a C library that implements the SVM API calls
  - Implements decoupled execution model of scalar CPU, stream processor running on different threads
  - Monitors memory transfer times between nodes
- Each Node is composed of many threads:
  - Scalar CPU thread
  - Stream Unit thread (runs kernels)
  - DMA thread (runs stream memory operations)
  - Dispatcher thread (handles out-of-order launch of instructions)

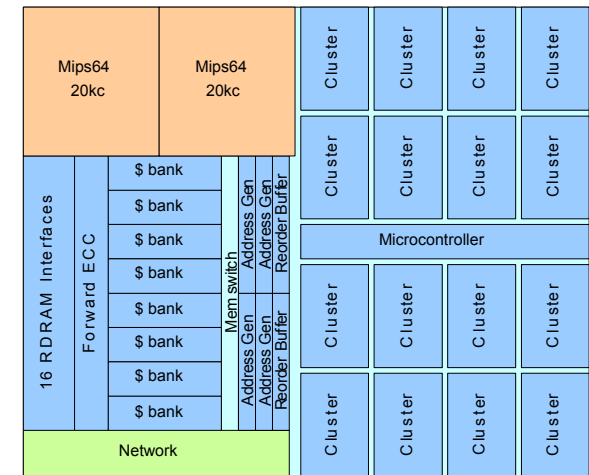
## Multi-node Simulation

- Perform small multi-node simulations using the accurate node simulator
  - Feasible for up to 4-8 nodes
  - Results in fairly accurate performance measurements
  - For more nodes, cycle-accurate simulations will take a prohibitive amount of time
- Larger scale simulations with 100s or 1000s of nodes will be done with the SVM simulator
  - SVM simulator will get kernel timings directly from schedules
  - Memory timings calibrated by comparisons with the accurate node simulator

## Simulation Flow

- Will use different tools for simulations at different resolutions
  - Functional simulator to develop applications
    - No timing information, just check for correctness
  - Cycle-accurate node simulator for accurate timings of small-scale simulations
    - Accurate timing measurements, but simulations take a long time to run
  - SVM simulator for large scale simulations
    - Uses parameterized machine model
    - Timings calibrated by cycle-accurate simulator

## Merrimac Node Simulator



- Cycle-accurate simulation of stream hardware (in blue)
  - Clusters
  - Microcontroller
  - SRF
  - Memory system (caches, AG / ROB)
- Execution-driven simulation of scalar hardware (in orange)
  - GNU MIPS simulator
  - Single instruction/cycle
  - 1-cycle latency pipeline for non-memory instructions
  - Memory operations go through stream memory simulator for unified stream/scalar memory system
  - Applications compiled using MIPS cross-gcc
- Network is simple throughput/latency model; routers etc. aren't simulated