

# A Near Memory Processor for Vector, Streaming and Bit Manipulations Workloads

---

**Mingliang Wei**, Marc Snir, Josep Torrellas (UIUC)  
Brett Tremaine (IBM)



Work supported by  
HPCS/PERCS

# Motivation

---

- Many important applications are not supported well on commodity processors
  - Have data-level parallelism (Vector, Stream)
  - Have poor temporal locality (Vector, Stream)
  - Demand high bandwidth (Vector)
  - Manipulate bits (Bit manipulation)
- Caches are not effective
- Some available parallelism is not exploited



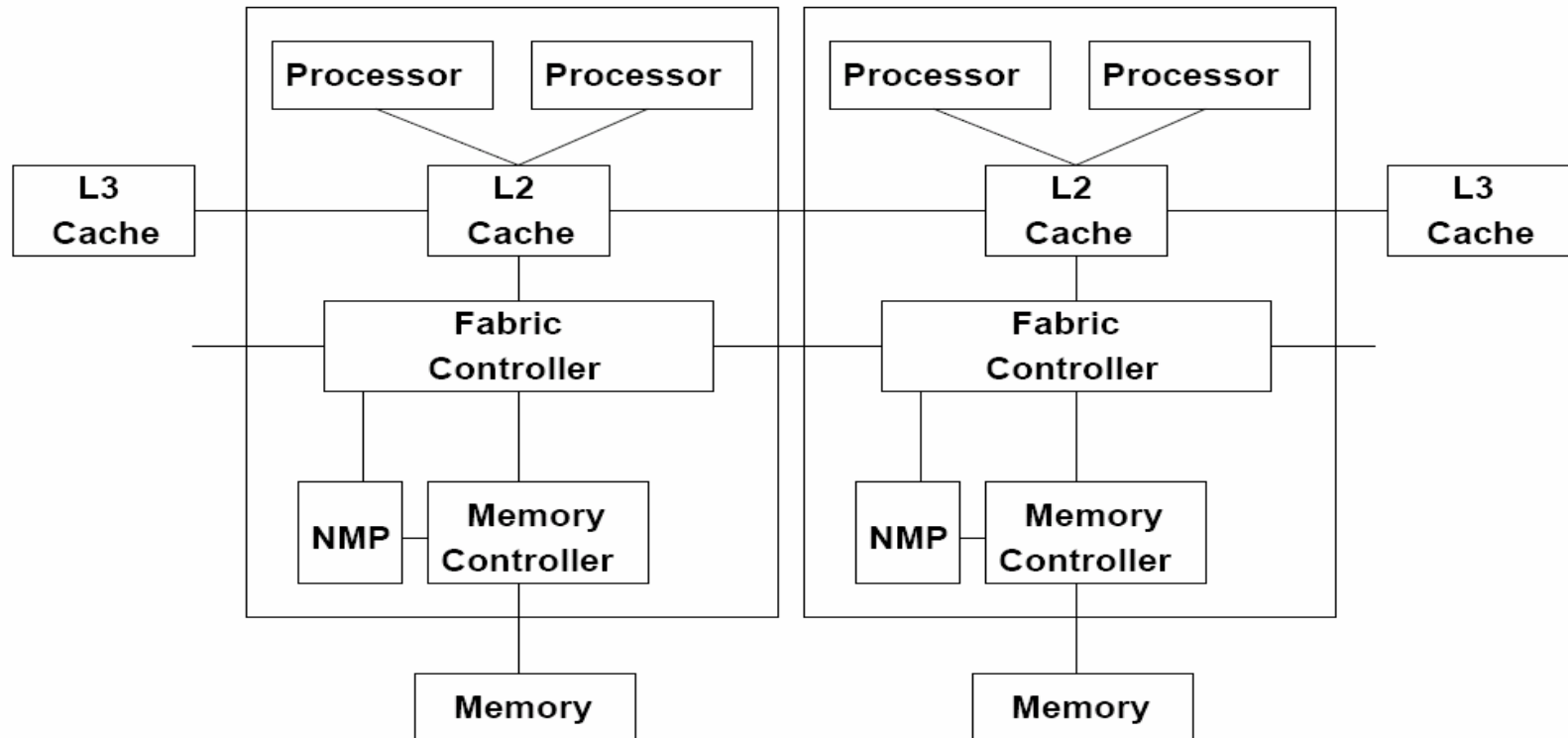
# Near-Memory Processor (NMP)

---

- Simple multithreaded processor with vector/stream/bit manipulation support
  - Integrated as coprocessor in main processor chip
  - Provides high performance for challenging apps.
    - Data parallelism, poor temporal locality, bit manipulation
  - Simple, general purpose, easy to program



# System Architecture



# Main Contributions

---

- Architecture that integrates vector, streaming and blocked multithreading
- Unified fast storage (scratchpad) for vectors and stream buffers
- Hardware supported dynamic scheduling of stream kernels



# Vector, Streaming and Multithreading

---

## ■ Vectors

- ↑ Have parallelism
- ↑ Memory latency hiding
- ↓ Large register set requirement

## ■ Streams

- ↑ Have parallelism
- ↑ Exploit producer/consumer locality
- ↓ Large register set requirement
- ↓ Compiler support for scheduling

## ■ Blocked Multithreading

- ↑ Simple implementation
- ↑ Tolerate variability in access latency of vector elements
- ↑ Tolerate variability in relative speed of communicating streams/threads
- ↑ Increase resource utilization
- ↓ Multiple contexts alive in the processor, expensive to implement if context is large



# Vector, Streaming and Multithreading

---

## ■ Vectors

- ↑ Have parallelism
- ↑ Memory latency hiding
- ↓ Large register set requirement

## ■ Streams

- ↑ Have parallelism
- ↑ Exploit producer/consumer locality
- ↓ Large register set requirement
- ↓ **Compiler support for scheduling**

## ■ **Blocked Multithreading**

- ↑ Tolerate variability in access latency of vector elements
- ↑ Tolerate variability in relative speed of communicating streams/threads
- ↑ Increase resource utilization
- ↑ Simple implementation
- ↓ Multiple contexts alive in the processor, expensive to implement if context is large



# Vector, Streaming and Multithreading

## ■ Vectors

- ↑ Have parallelism
- ↑ Memory latency hiding
- ↓ Large register set requirement

## ■ Streams

- ↑ Have parallelism
- ↑ Exploit producer/consumer locality
- ↓ Large register set requirement
- ↓ Compiler support for scheduling

## ■ Blocked Multithreading

- ↑ Tolerate variability in access latency of vector elements
- ↑ Tolerate variability in relative speed of communicating streams/threads
- ↑ Increase resource utilization
- ↑ Simple implementation
- ↓ Multiple contexts alive in the processor, expensive to implement if context is large



**Scratchpad**



# Scratchpad

---

- High-bandwidth, fast local memory
- Stores scalars, vectors and stream buffers
- Shared by all threads
- Not saved on context switch
- Full/Empty bits for thread synchronization



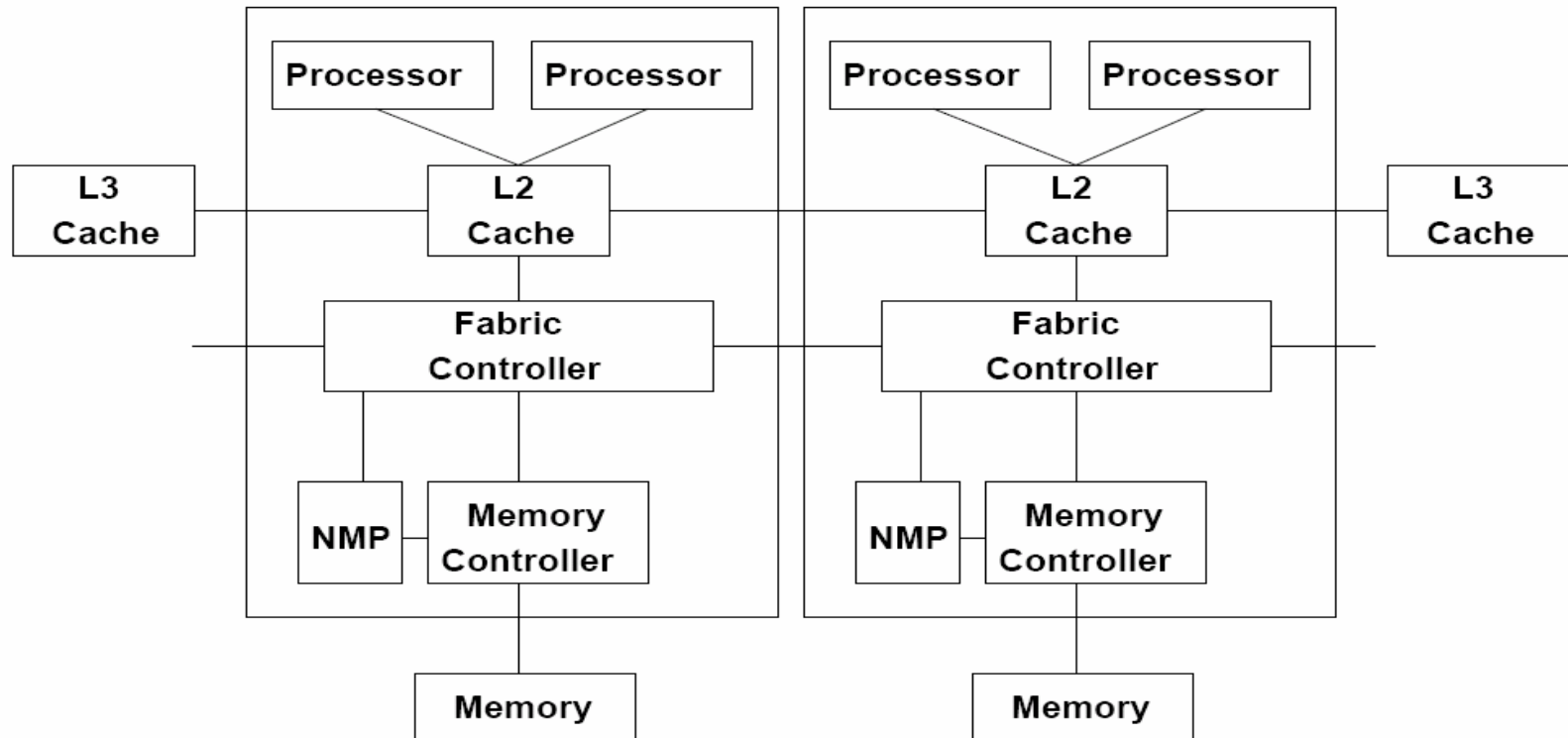
# Blocked Multithreading

---

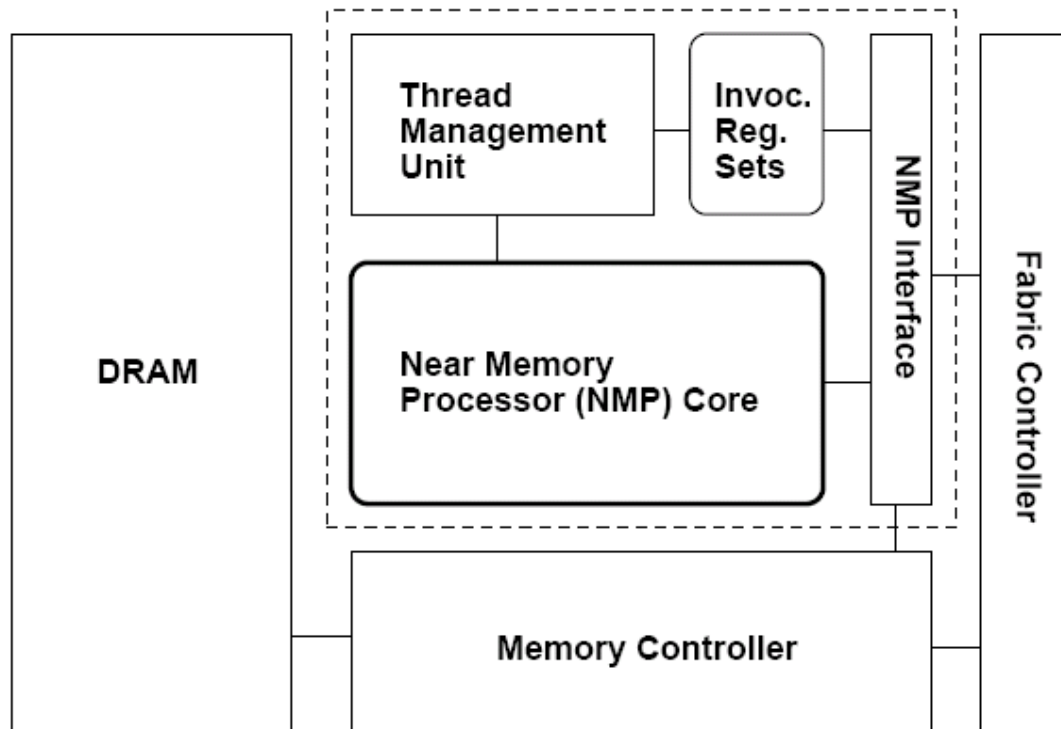
- Context switch when processor expects long idle time
  - Memory access
  - Stalled on synchronization
- Do not save much state on context switch: Scratchpad is not saved



# System Architecture



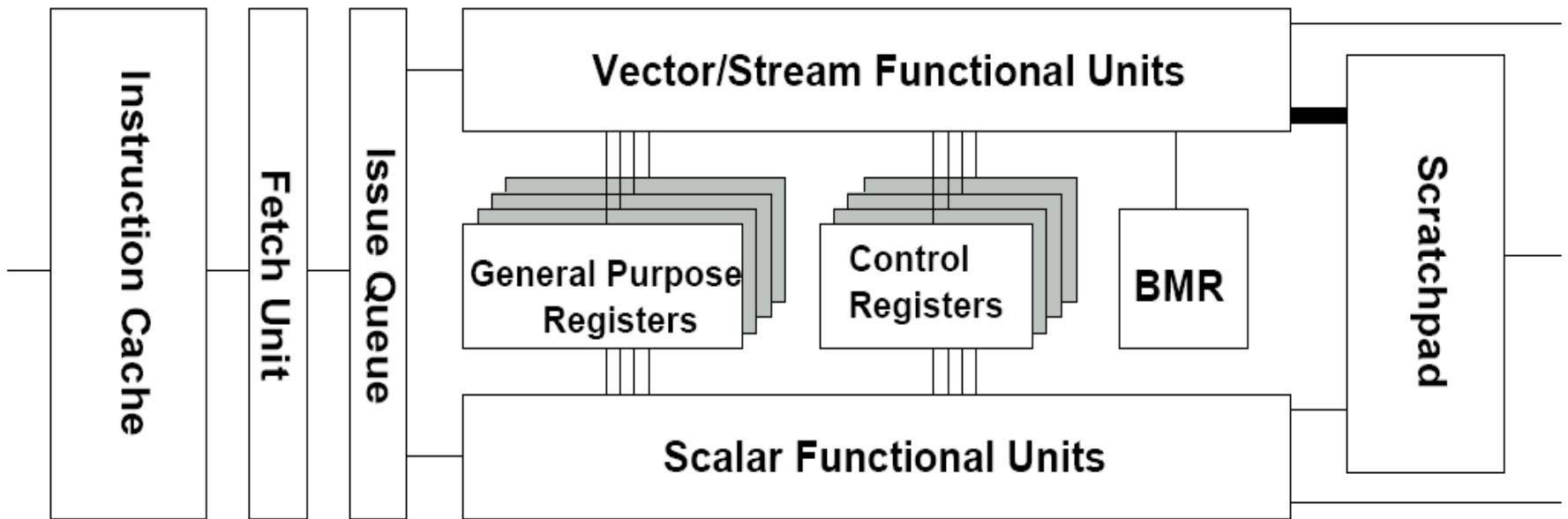
# NMP Architecture



- In-order core

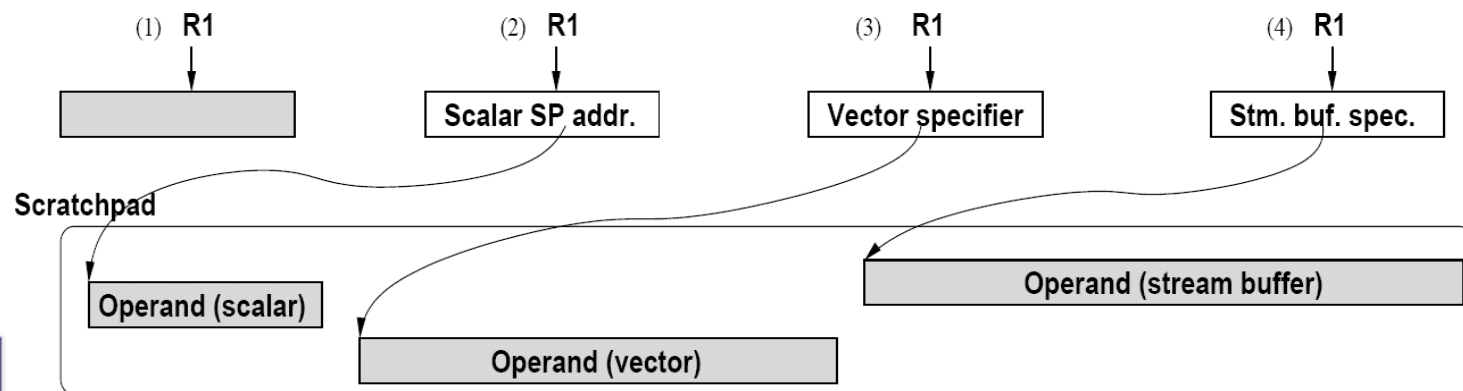
# Core Architecture

---



# Data Organization in the Scratchpad

- Addressing modes
- 3 bits for each byte: F/E (efficient synch.), error, mask
- Addressed with virtual addresses
- May suffer page misses. Pages are lazily paged out to mem (special pager)



# Specifiers Format

## ■ Vector Specifier

0000 ... .. 0000	Number of entries	Log # bytes per element	Start address
29 bits	12 bits	3 bits	20 bits

## ■ Stream Buffer Specifier

Read only flag	0000 ... .. 0000	Head of the buffer	Number of entries (capacity)	Log # bytes per element	Start address
1 bit	16 bits	12 bits	12 bits	3 bits	20 bits

## Consumer Specifier

0000 ... .. 0000	Tail of the buffer	Number of entries (capacity)	Log # bytes per element	Start address
17 bits	12 bits	12 bits	3 bits	20 bits

## Producer Specifier



# NMP ISA

---

- Simplified load/store ISA +
  - Vectors and stream buffers supported via additional addressing modes
  - Additional vector load/store instructions
  - Bit Manipulation
    - Bmm, Leadz, Popcnt, Sshift, Mix



# Other Issues

---

- Asynchronous interface to main processors
  - User-level NMP thread invocation
  - NMP interface mapped to user space
  - Invocation completion set memory flag
- Exceptions in NMP
  - Virtual memory exceptions: Not precise but restartable and handled in SW
  - Error bit
  - Vector and stream buffers in Scratchpad cannot cross pages



# Programmer's view on NMP threads

---

- NMP uses virtual addresses to access:
  - local scratchpad (short addresses)
  - main memory and remote scratchpads (64-bit addresses)
- Main processor uses virtual addresses to access:
  - All scratchpads and main memory (64-bit addresses)
- Cooperating threads on the same NMP communicate and synchronize via scratchpad



# Programming Environment

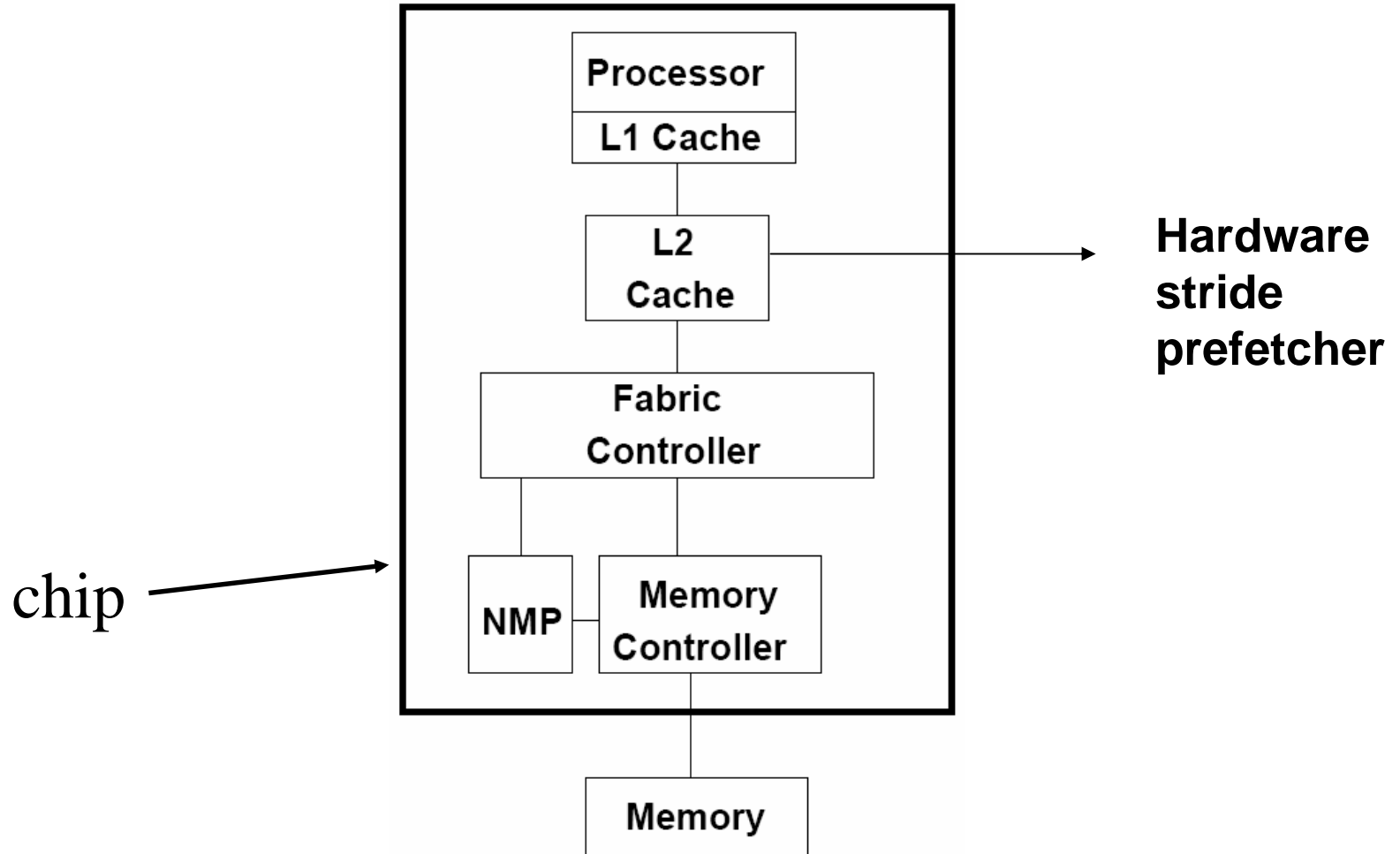
---

- Libraries to connect, disconnect and invoke in user-level the NMP
- Possible Programming Support
  - Hand-coded libraries for special applications\*
  - Libraries separately compiled by vectorizing compiler
  - Compiler that splits code into main-processor part and NMP part



\* Only level supported so far

# Architecture Evaluated



# Simulation Parameters

---

## ■ NMP

- ❑ 4G Hz
- ❑ In-order
- ❑ Issue width: 2
- ❑ # of pending memory load/store: 128/128
- ❑ Max # of contexts: 4
- ❑ # of Lanes: 16

## ■ Main Processor

- ❑ 4G Hz
- ❑ Out-of-order
- ❑ Issue width: 4
- ❑ # of pending memory load/store: 16/16
- ❑ Hardware stride prefetcher



# Benchmarks Evaluated

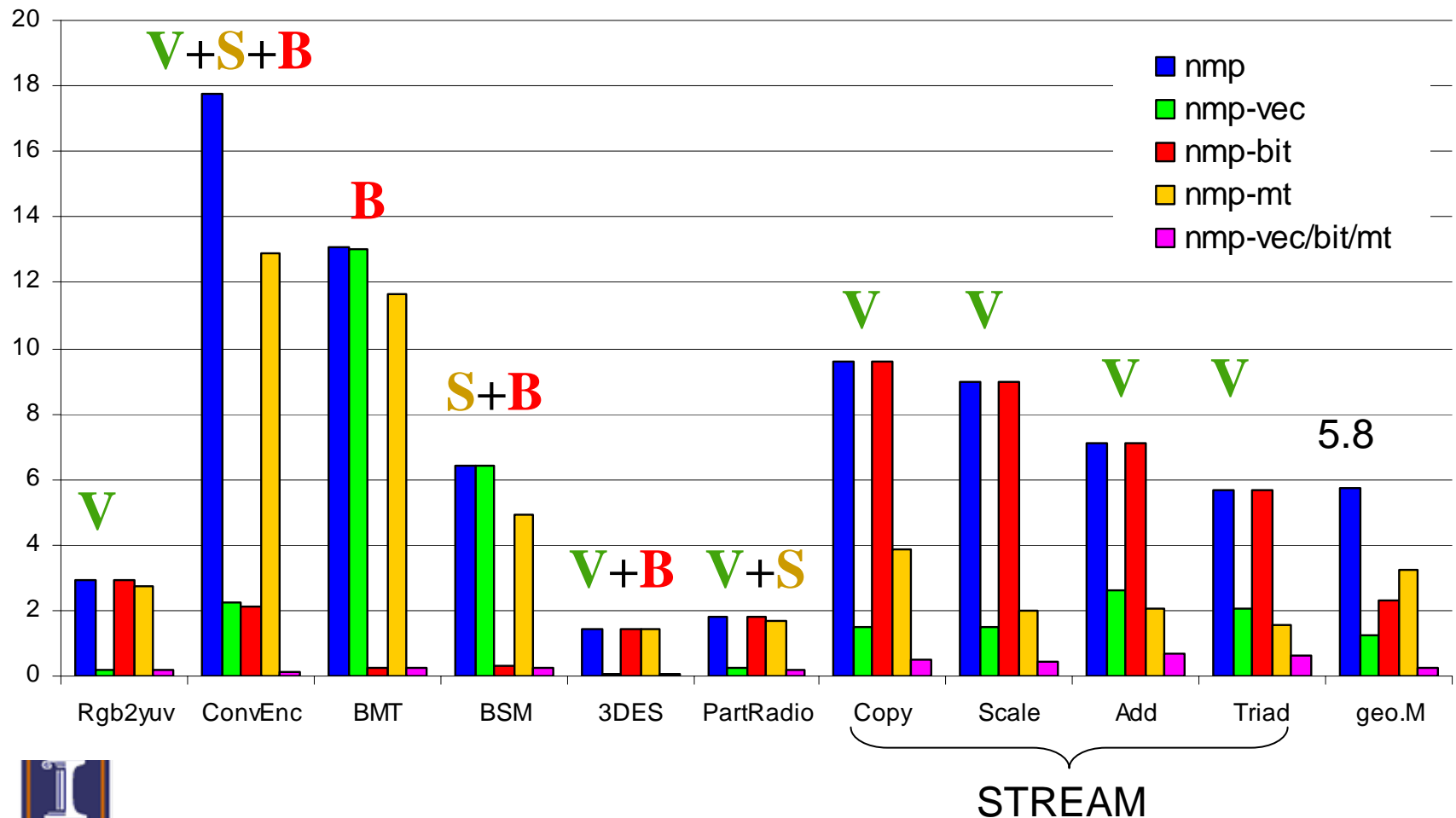
---

Application	Vector?	Stream?	Bit Manip?	# Threads	Remarks
Rgb2yuv	X			4	Convert the RGB presentation to YUV
ConvEnc	X	X	X	3	Convolutional encoder
BMT			X	4	Bit matrix transposition
BSM		X	X	3	Bit stream manipulation
3DES	X		X	4	3DES encryption
PartRadio	X	X		3	Partial radio station
Stream	X			4	Simple vector operations

- Representative for vector, streaming and bit manipulation applications
- Simple kernels (easy to hand-code)



# NMP Speedups Over Main Processor



# Related Work

---

- Inspired by multithreaded vector architecture (Espasa & Valero, HIPC'97, HPCA'97)
- Processor in memory provides high memory bandwidth
  - DIVA (Hall et al, SC'99) , FlexRAM (Kang et al, ICCD'99), Active Pages (Oskin et al, ISCA98)
  - Expensive production cost
- Intelligent memory controller can better utilize the memory bandwidth
  - Impulse (Zhang et al, 2001). No computation off-load
- Stream processors (Dally et al, SC'03, Khailany et al, 2001)
  - No hardware supported multithreading
- CELL processor (Pham et al, ISCC'05)
  - Multiple simple SIMD engines vs wider more complex single SIMD engine



# Conclusion

---

- Selected coprocessor design points:
  - Off-load vector, streaming, bit manipulation ops
  - Simple/compact design
  - Fairly general purpose core
- Demonstrate high speedups (5.8 average) in many important, memory intensive applications



# Future Work

---

- Demonstrate ease of programming:
  - Programming interface
  - Compiler support
- Demonstrate performance on complete applications
- Study variations in the NMP design



---

# BACKUP



# Bit Manipulation Instructions

---

Instruction	Remarks
Leadz	Count the leading zeros of a scalar.
Popcnt	Count the number of ones in a scalar.
Bmm_load	Load the $64 \times 64$ -bit matrix from the scratchpad into the BMR. It is a special vector load instruction (A regular vector load instruction transfers data between the scratchpad and the main memory.).
Bmm	Bit multiply the source operand with the matrix in the BMR.
Sshift	Logic left- or right-shift a block of data, e.g., 128 bytes. The shift can be rotational or not rotational. In the latter case, zeros are shifted into the block.
Mix	Bit-interleave higher(lower) half of two words.



# Simulation Parameters

NMP Parameters	
Parameter	Value
Frequency	4GHz in-order
Issue Width	2
# Scalar FUs	1Int FU, 1FP FU
# Vector FUs	1Int FU, 1FP FU
# Lanes	16
# Pending Memory Ops (Ld, St).	128, 128
# Contexts	4
Time to Context Switch	4 cycles
Policy for Context Switch	Switch after 20 idle cycles

Memory Parameters	
Parameter	Value
L1, L2, Scratchpad size	32KB, 1MB, 64KB
L1, L2 associativity	2-way, 4-way
L1, L2 line size	64B, 64B
Main proc. to L1, L2, memory round-trip latency	2, 10, 500 cycles
NMP to Scratchpad, memory latency	6, 470 cycles
Bandwidth b.t. vec. units and Scratchpad, Scratchpad and memory	256GB/s, 32GB/s

Main Processor Parameters	
Parameter	Value
Frequency	4GHz out-of-order
Fetch Width	8
Issue Width	4
Retire Width	8
ROB size	152
I-window size	80
Int FUs	3
FP FUs	3
Mem FUs	3
Pending Ld/St	16, 16
Branch Pred.	Like Alpha 21464
Branch Penalty	14 cycles
Hardware Prefetcher	16-stream stride
Prefetch Buffer	16KB
Pref. Buf. Hit Delay	8 cycles



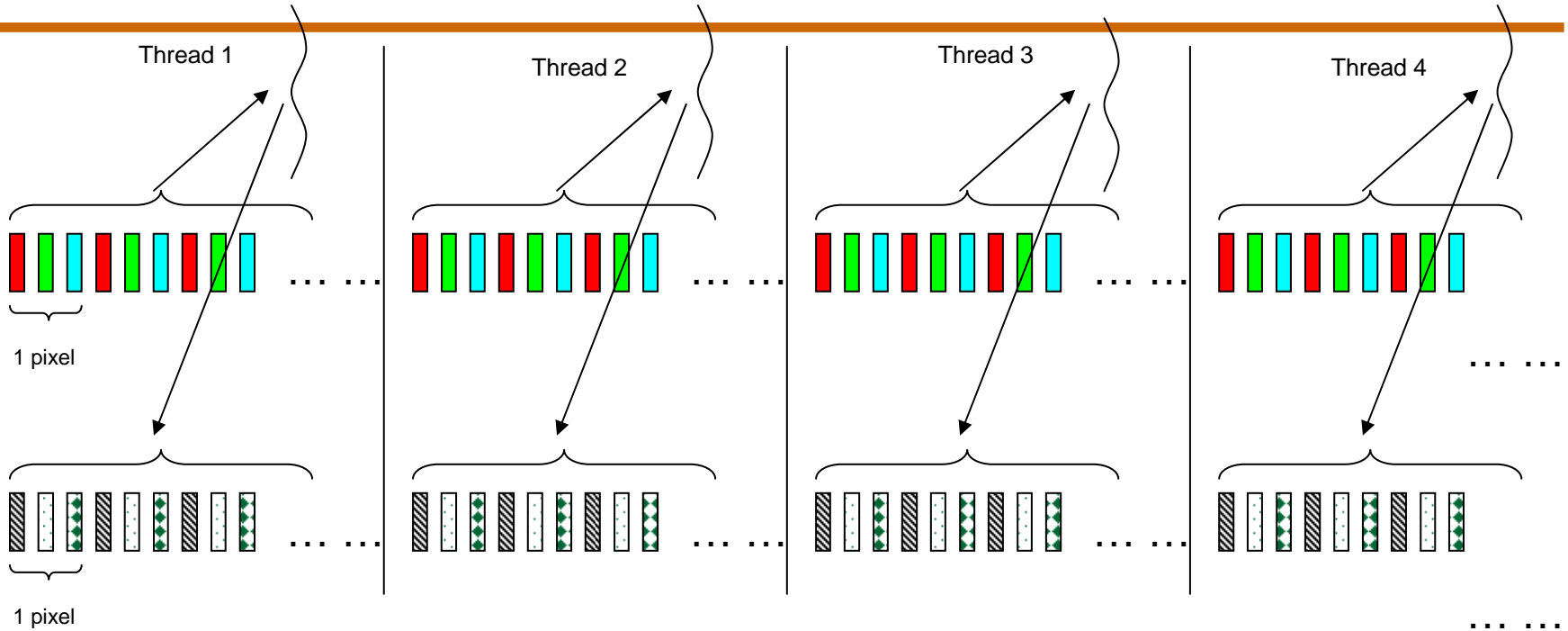
# Synchronization between NMP and Main processors

---

- Main processor
  - Invoke NMP: mem. mapped reg.
    - OS select which NMP to invoke
    - Invocation Packet (func, param, affinity\_data, completion\_flag, SP\_size)
    - write *IP reg*
    - clear *Ready Reg*
  - Synchronize: polling on cf
- NMP
  - Execute threads: take IP from reg, start a thread, set *Ready Reg*.
  - Thread terminates: set cf



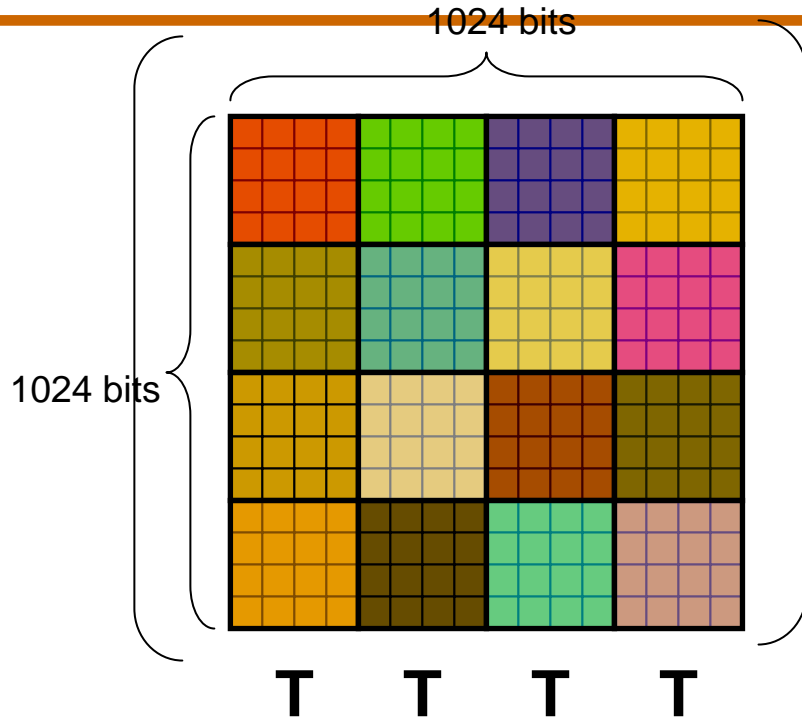
# RGB2YUV (Vector)



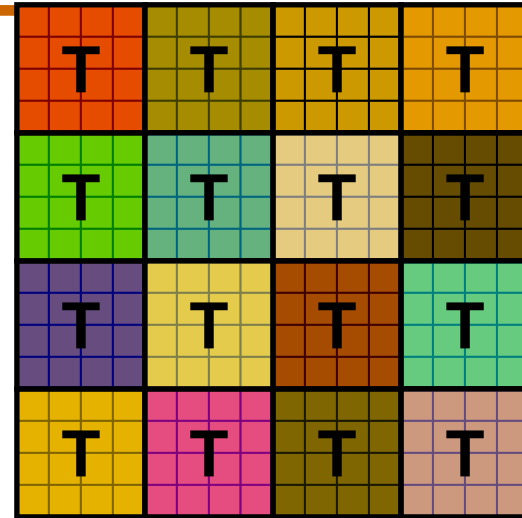
- Input: 1000 x 200 pixels, output is the same size
- Conversion of each pixel is independent
- 4 threads, each processes a partition of the input
- Each thread processes 16 pixels in parallel



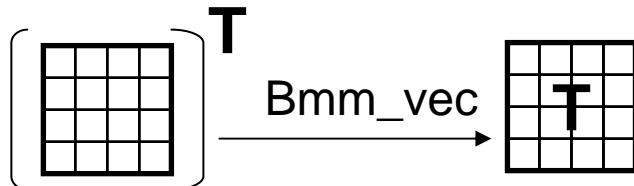
# BMT (Bit manip)



T



- Transpose 4 1024x1024 bit matrices
- 4 threads, each trans. 1 matrix
- Threads use:
  - Vector load to load data
  - Bmm\_vec to trans. a tile
  - Vector store



A 64x64 bit tile



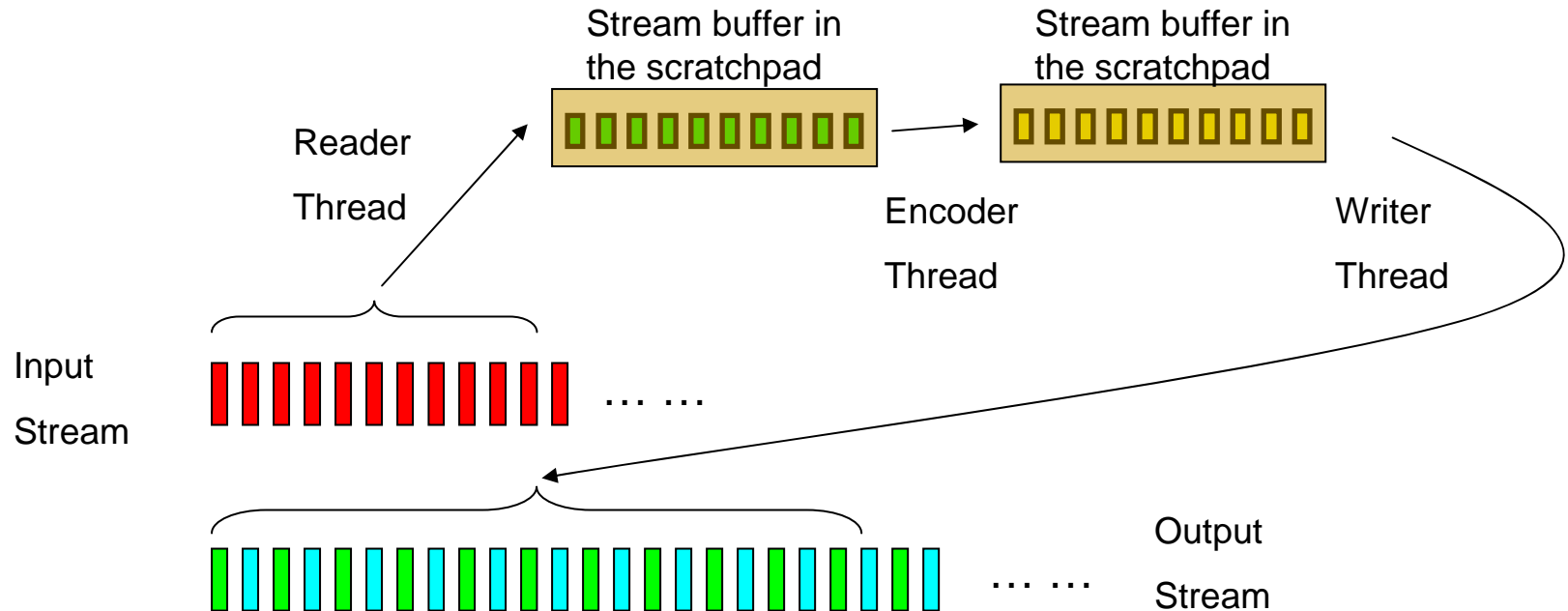
# 3DES (Vector)

---

- Input: set of 64-bit word values. Output is same-size encrypted word values.
- Vectorized the load/store and the computation
- Counter mode (more parallelism)
- 4 threads, each work on a partition of the input (like Rgb2yuv)
- Each thread processes 16 elements in parallel
- Bmm to perform initial and final permutation (though a small fraction of total execution time)



# ConvEnc (Vector + stream + bit)



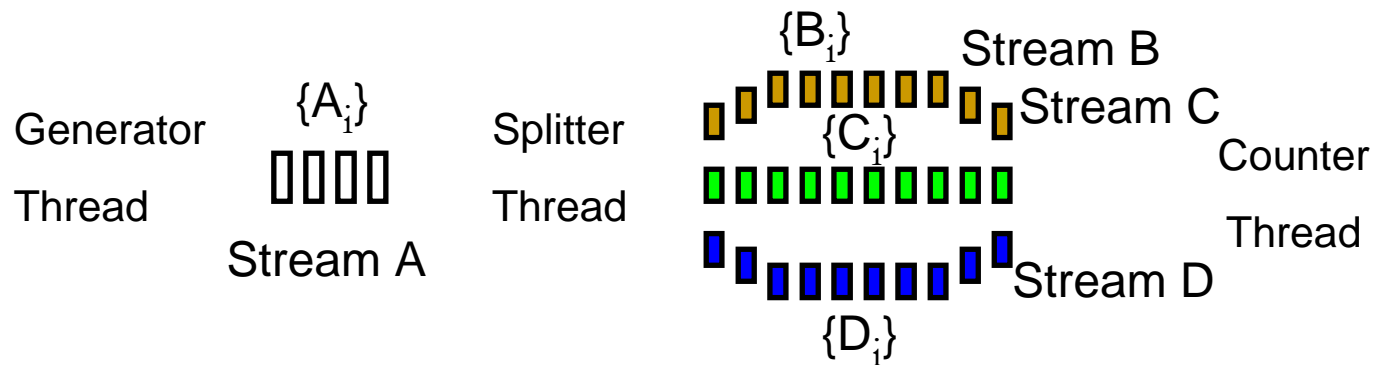
- Input: bit stream; output: 2x bit stream
- Encode 64 bytes at a time
- Sshift instruction to shift 64 bytes of data
- Mix instruction to interleave two blocks of bits



# BT (Stream + bit manip)

Generate  $\{A_i\}$   
 $\{B_i\}$  = take 5, drop 6, take 5 ... from  $\{A_i\}$   
 $\{C_i\}$  = drop 5, take 5, drop 6, take 5, drop 6 from  $\{A_i\}$   
 $\{D_i\}$  =  $(C_i \text{ or not}(B_{i+1})) \text{ xor } (\text{not}(C_i) \text{ or } B_{i-1})$   
 $\{E_i\}$  =  $D_i \text{ xor } D_{i+37} \text{ xor } D_{i+100}$   
 Identify sequences of 0s in E that are longer than 100

- Three threads in the NMP: Generator, Splitter, Counter
- Bmm to split the  $\{A_i\}$



# PartRadio (Vector + stream)

---

- Input: a float-point stream. Output is a float-point stream
- 3 threads (like ConvEnc)
  - Low pass filter
  - Demodulator
  - Equalizer
- Thread processes 16 elements in parallel (vector)

