

# Advanced Computer Systems Architecture

## Chip-Multiprocessors: Applications and Architectures

CSE 561M

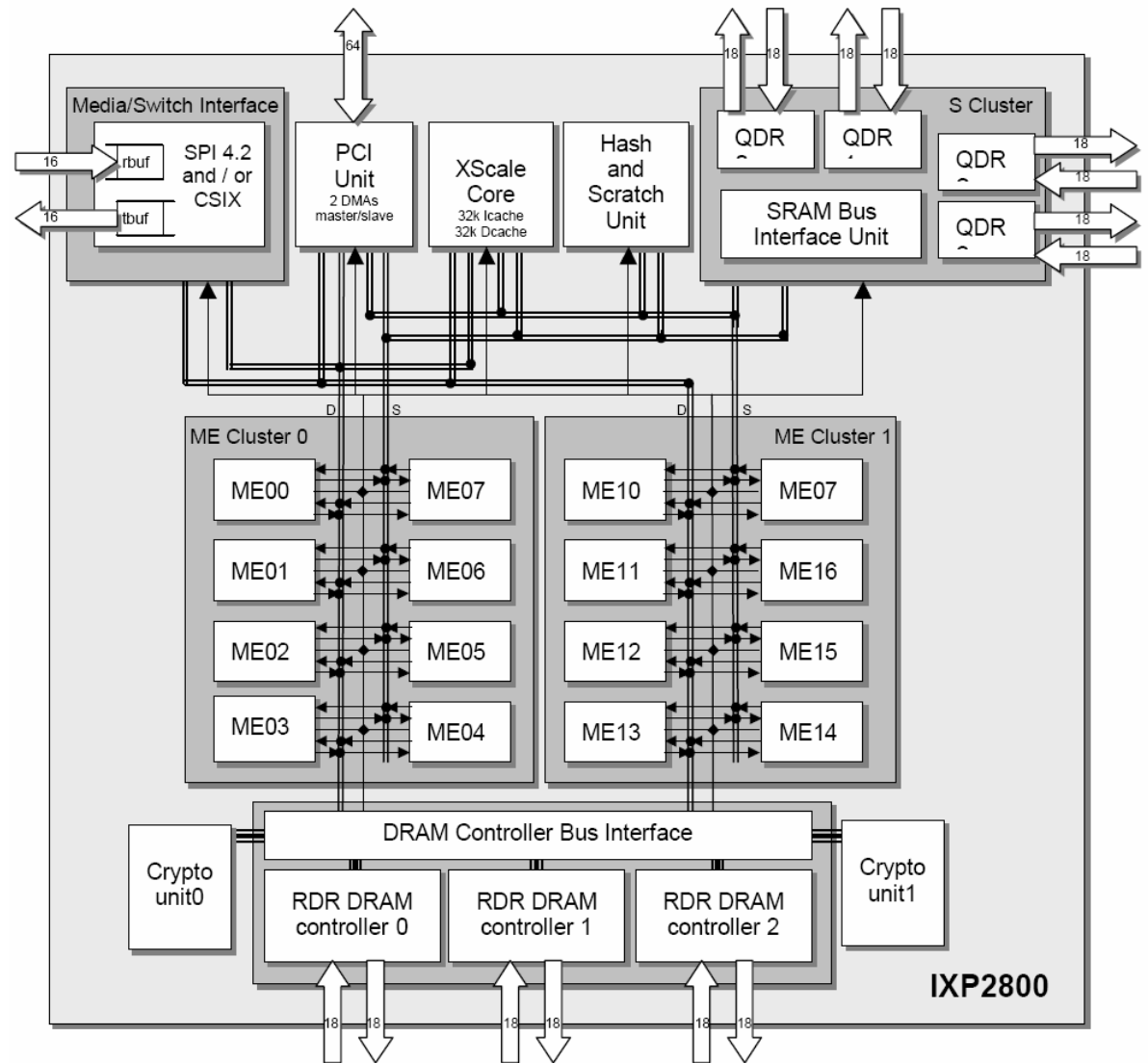
Prof. Patrick Crowley

# Plan for Today

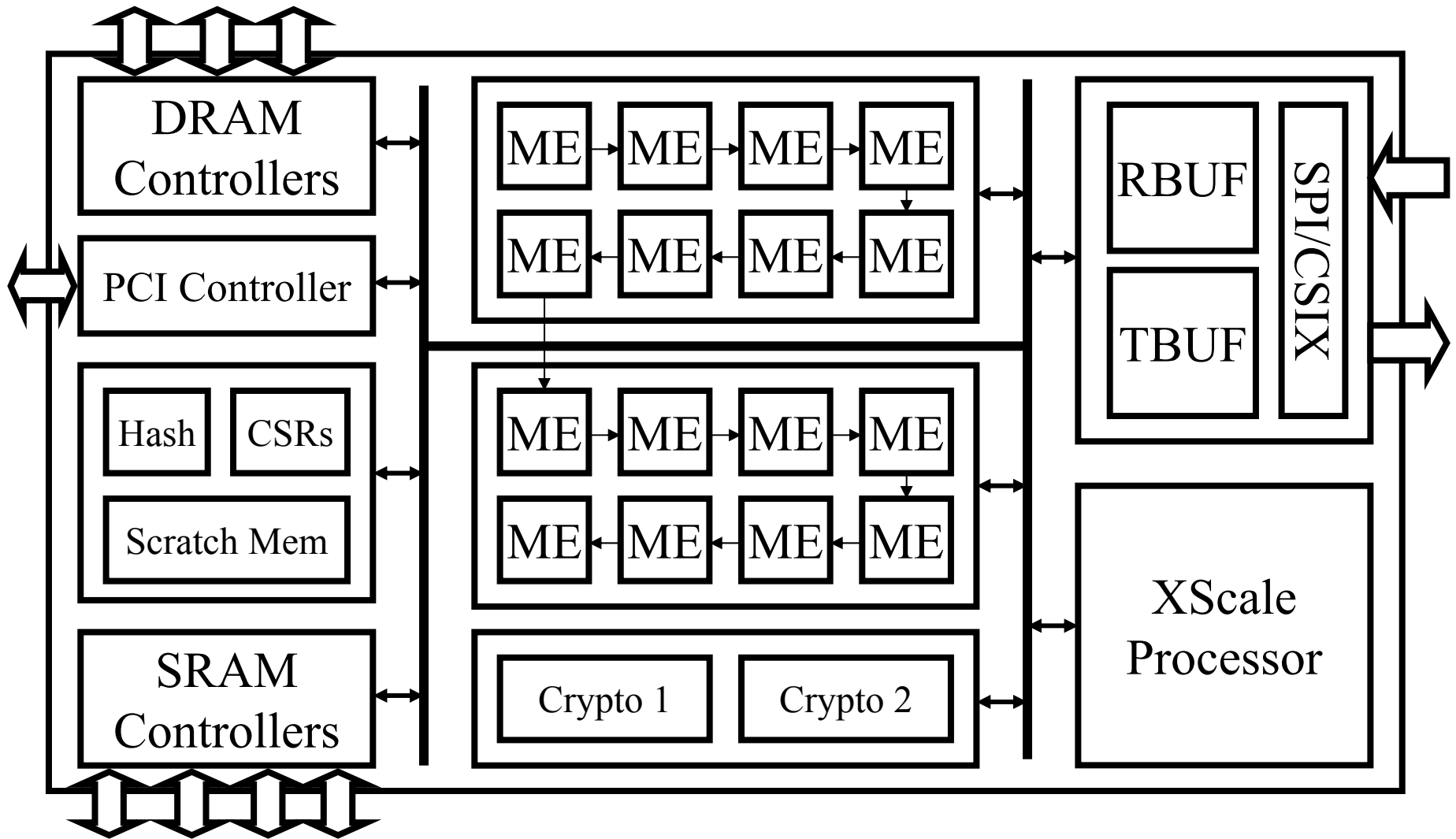
- Questions
- Today's discussion
  - IXP Structure
  - Basic IXP operation

# IXP2800 Organization

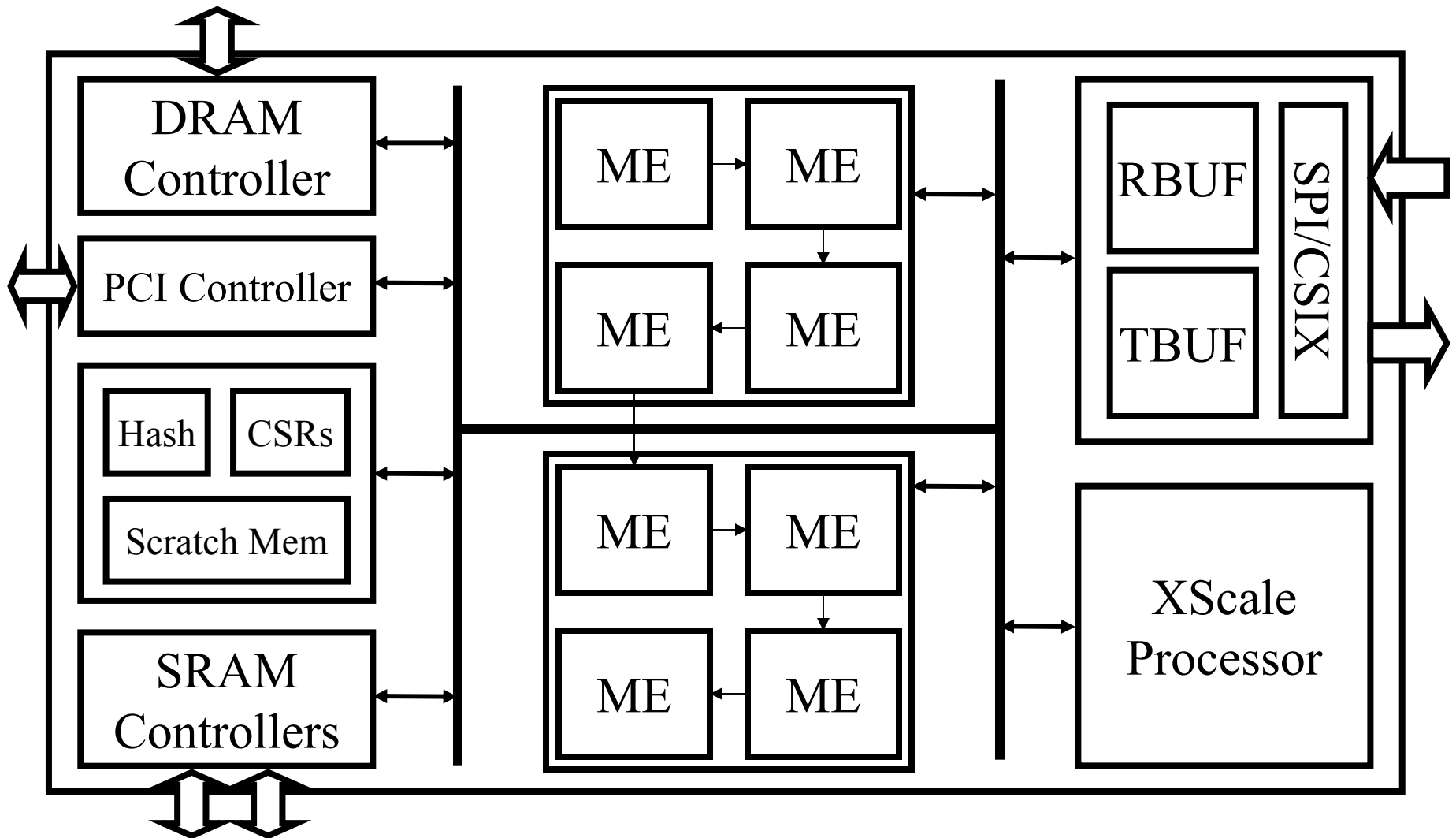
- 17 processors
- Hardware assists
- 10 “clusters”
- Multiple clocks
  - 1.4 GHz
  - 700 MHz
  - 200 MHz
  - 133 MHz



# IXP 2850 Block Diagram



# IXP 2400 Block Diagram



# Cluster-based Organization

- Each cluster of microengines has its own
  - Command bus
  - SRAM bus
  - (Although figures are a bit ambiguous)
- Clusters + duplicate buses = reduced contention
  - Load balance across clusters

# Media-Switch-Fabric Interface (MSF)

- Connects IXP to physical I/O interfaces
  - SPI-4 (chip-to-chip), CSIX-L1 (for switch fabrics)
  - Can multiplex both simultaneously
- 10 Gb/s and 15 Gb/s in and out (or vice versa)
- Coming and going packets staged in receive and transmit buffers (RBUF, TBUF)
- Packets broken into chunks, called *mpackets*, which we will study in detail later
- MSF provides a programmer interface for packet reception and transmission

# IXP Chassis

- Multiple buses interconnect IXP clusters
  - Connects microengine *transfer registers* to other resources
- Separate command and data buses
  - Similar to split transaction, but more sophisticated
  - Commands are like messages, notifying a resource of a pending service request
- Chassis was designed first

# XScale Processor

- Implementation of ARM version 5TE
- Has traditional memory system
  - 32KB I and D caches
  - Virtual memory and MMU
- 700 MHz
- Manages the chip, used to handle “exceptional” conditions
- Runs Linux or real-time OS such as VxWorks



# SRAM

- 4 independent quad data rate (QDR) SRAM controllers
  - Each channel 200 MHz, 32b wide
  - Up to 64 MB per channel
  - Addressing
    - Logical width: 4 bytes (only access addresses 0, 4, 8, ...)
      - Same as local memory and scratchpad
    - Addresses across channels do not overlap
      - Each channel gets only a fraction of the physical address space

# SRAM, cont'd

- Supported Operations
  - Traditional random reads/writes
  - Atomic operations for
    - Bit-test-and-set, Bit-test-and-clear, Bit-test, Bit-clear, Add, Subtract, Test-and-add, Test-and-clear, Swap, Increment, Decrement
  - Linked-list queue and dequeue
  - Circular buffer (i.e., ring) inserts and deletes

# DRAM

- 3 independent Rambus DRAM controllers
  - Each channel 133 MHz, 16b wide
  - Up to 1 GB per channel (up to 2 GB total)
- Controllers allow data to move directly from MSF to DRAM (i.e., not through an ME)
- Addressing
  - Logical width: 8 bytes (only access addresses 0, 8, 16, ...)
  - Addresses are interleaved, or striped, across channels; implemented in hardware

# Question

- Why two off-chip memory types?

# Cryptography Units

- 2 identical units on the IXP2850
- Implements bulk encryption via
  - Advanced encryption standard (AES)
  - Triple Data Encryption Standard (3DES)
- Authentication
  - Secure Hash Algorithm (SHA-1)
    - Computes a one-way hash over input data (i.e., message digest)
  - Hashed Message Authentication Code (HMAC)
    - Keyed message digest over input data; also provides integrity check (i.e., data didn't change in transit)
- Checksum accumulator

# SHaC Unit

- Scratchpad memory
  - On-chip 16KB memory, with all the SRAM operations
  - 700 MHz
- Interface to XScale external peripherals and timers
- Hash unit
  - 48-, 64- or 128-bit hashes (via programmable polynomial division)
- CAP unit
  - Interface to all control status registers (CSRs) on-chip
  - Implements thread signaling, register reflection (allows threads to write/read other transfer registers), and ME/XScale interrupts

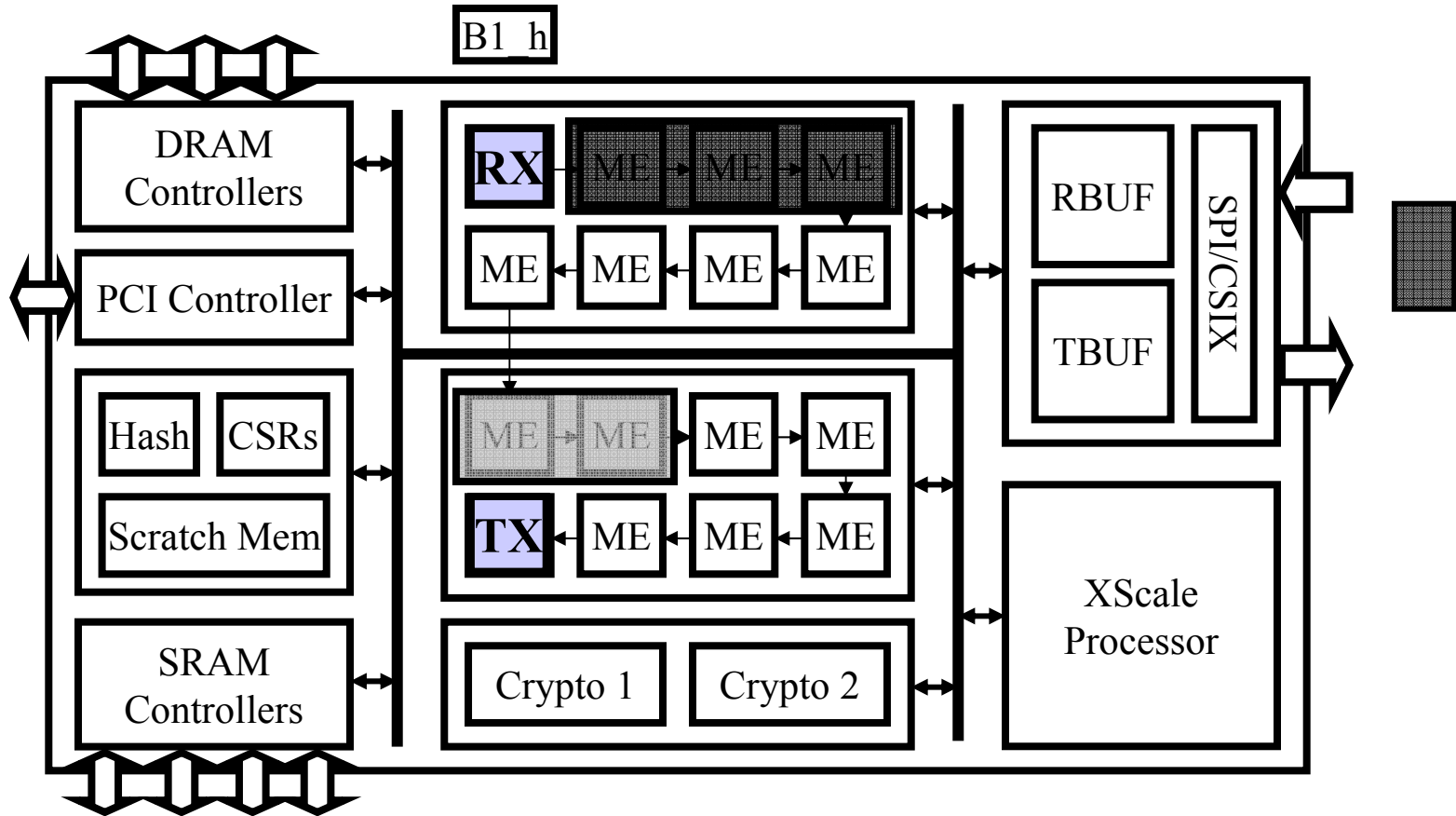
# PCI Unit

- Interface to vanilla PCI interface, to connect to a host processor or peripheral devices
- Can act as either PCI Master or Target
- Can efficiently perform DMA bulk transfers to/from IXP SRAM or DRAM and external device across PCI bus

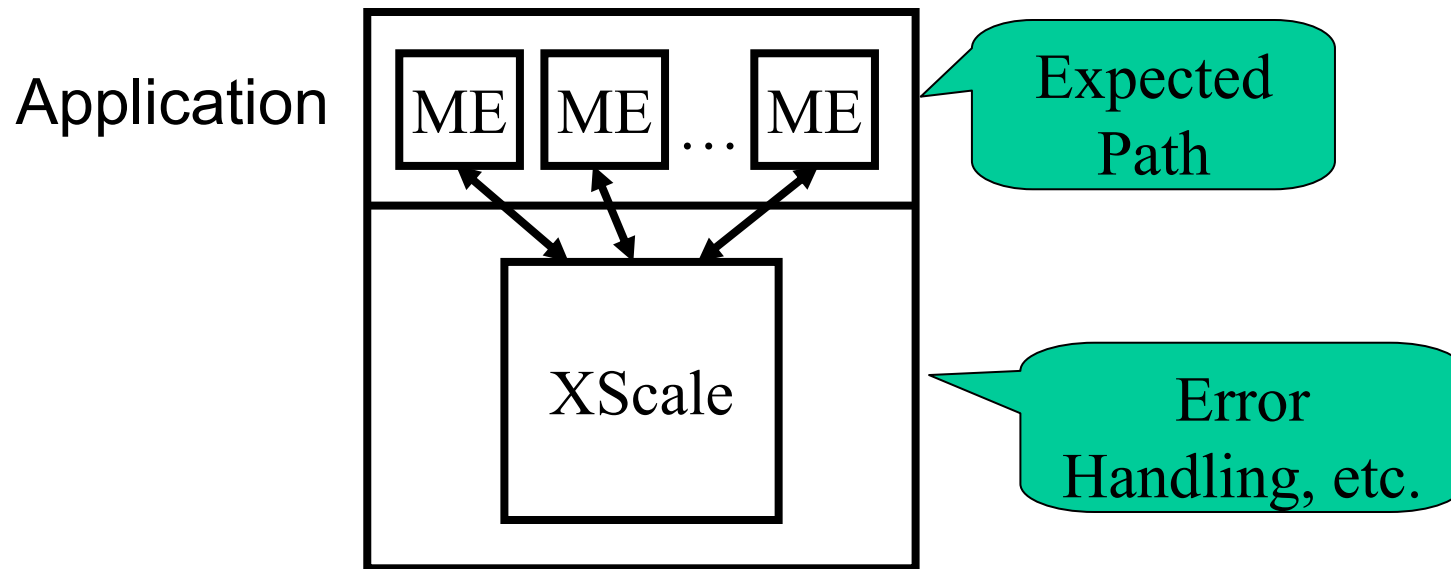
# Basic IXP Operation

- Simple packet handling happens in three logical stages
  - Receive
  - Process
  - Transmit
- Each pipeline stage is implemented on a different microengine (ME)
- On-chip rings are used for inter-ME transfer

# IXP Operation Illustrated

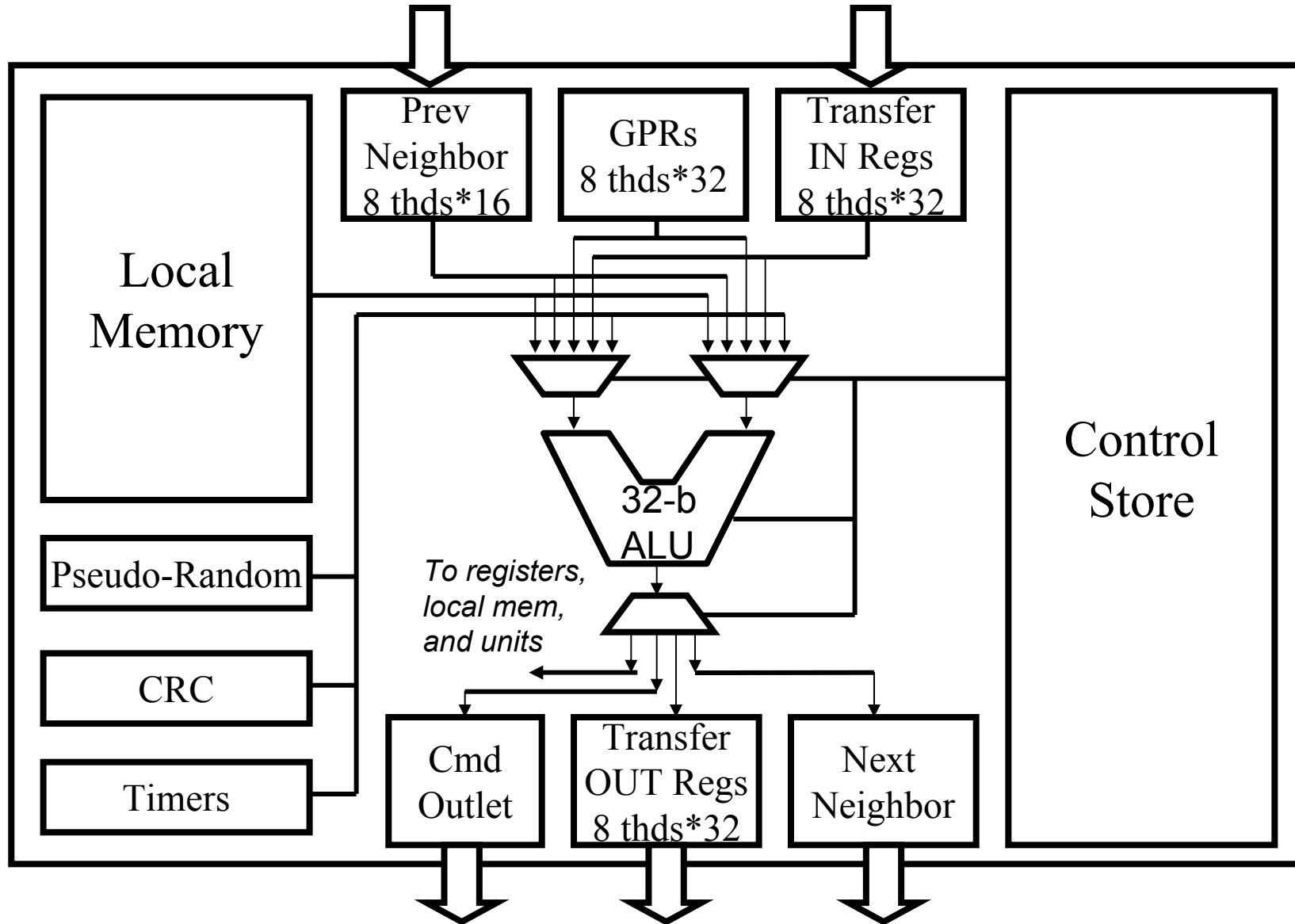


# IXA/IXP Programming Model



- Separate expected path from complete program
  - MEs execute expected path
  - Only execute error handling, management, and control tasks on the XScale
  - Pass messages from one to the other
- Three reasons for this
  - MEs have limited control store (8K instructions each)
  - XScale hosts an operating system and network stack
  - Programmer needs to understand expected performance

# MEv2 Block Diagram



# ME Design Goals

- Space efficiency (high compute density)
- Fast clock
- Many registers
- Local memory
- Efficient inter-ME communication
- Tolerate memory latency

# ME Instruction Set

- Approximately 50 instructions
  - Arithmetic, logic (no F-P, no divide)
    - Bit, byte and word widths
  - Control flow
  - I/O instructions manipulating external resources such as SRAM, DRAM
    - Including SRAM ops discussed earlier
    - Instructions need not wait for results
  - Branch delay slots
  - CRC unit computes on 32 bit values

# ME Registers

- All 32b, all register files are banked
  - General-purpose registers – 256
  - Transfer registers – 512
  - Next-neighbor registers – 128
- Plus 640 words of local memory
  - Not registers, but accessed in similar manner

# Content Addressable Memory (CAM)

- 16 entries, 32b tag and 4-bit state
- Usage:
  - Lookup a 32b value
  - That value is compared against all 16 entries
  - 9b result is returned
    - 1 bit indicates hit/miss
    - 4 bits are the stored state
    - 4 bits indicate matched entry
  - On miss,
    - 4 bits indicate LRU entry

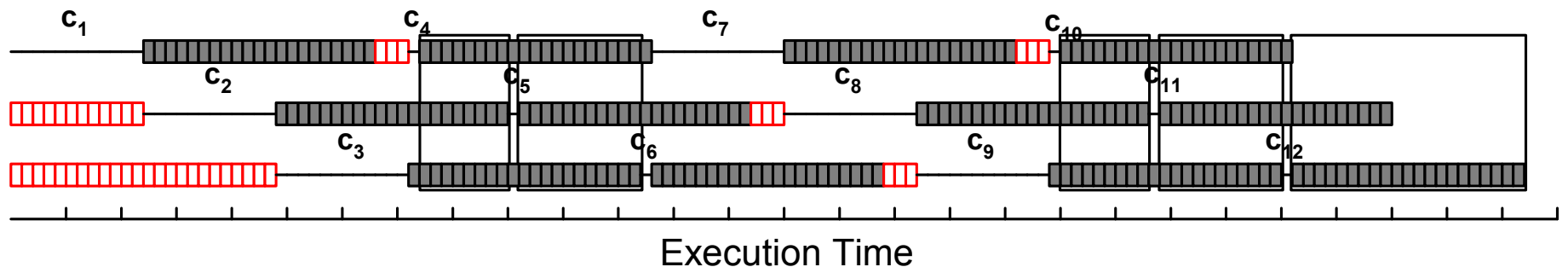
# ME Threads and Scheduling

- 8 HW-assisted threads of execution
  - Duplicate registers and control state
- HW-based arbiter maintains a round-robin schedule
  - Non-preemptive
- All execute from the same control store

# Question

- What is the point of multithreading?

# Sample Execution Timeline



- 3 threads each with 4 basic blocks
- Solid black line: *Compute* time
- Gray boxes: *Memory* time
- Red boxes: *Stall* time
- Background boxes: *Idle* time
- Completion time = sum of all compute and idle times

# ME Signals

- Each ME has 15 numbered signals
- When making an external request, an ME passes along a signal to be raised upon completion
- Control flow can be signal dependent
- Enables multiple outstanding references to *the same unit*
- In general, signals allow MEs to deal with resources asynchronously (big difference w.r.t. general purpose processors)

# Challenge

- Most features have both pros and cons
  - Multiple MEs, multiple memories, threads, etc.
- Challenge: how to solve important problems, cost effectively, with such a feature-rich, heterogeneous system
  - How much can compilers/tools can help is an open question

# Assignment

- Thursday:
  - **Commentary:** *The Push of Network Processing to the Top of the Pyramid*, Will Eatherton's keynote from ANCS 2005.
- Tuesday
  - Complete the IXP tools tutorial
  - Submit a **commentary** on the following:
    - Find (or write) a small piece of C code (it can be absolutely anything, completely trivial if you like) to drop into the tutorial project. Simulate that code, check that it works, and determine its run-time and microengine utilization. In your commentary, describe your program, include the source, and describe the performance results you gathered. If you had any strange problems that kept you from simulating, describe those. (Spend no more than 45 minutes doing this.)