INF5062:
Programming asymmetric multi-core processors

Introduction

1/9 - 2006

Overview

- ✓ Course topic and scope
- ✓ Examples of asymmetric processors
- ✓ (Very) short intro of the Intel IXP 2400 network processors

INF5062: The Course

People

- ✓ Morten Pedersen (TA)
  email: mortp @ ifi
- ✓ Carsten Griwodz
  email: griff @ ifi
- ✓ Pål Halvorsen
  email: paalh @ ifi

About INF5062: Topic & Scope

- ✓ **Content:** The course gives …
  - … an overview of asymmetric multi-core processors in general and network processor cards in particular (architectures and use)
  - … an introduction of how to program the Intel IXP 2400 network processors
  - … some ideas of how to use/program asymmetric multi-core processors
    (guest lectures and paper presentations)

About INF5062: Topic & Scope

- ✓ **Lab-assignments:**
  An important part of the course are lab-assignments where the students should make a program for the Intel IXP2400 network processor
  1. protocol statistics – download and run ***wwpingbump*** and then extend it to give processor, interface and protocol statistics
  2. packet bridge with ARP support – forward packet to correct interface (of 3 available)
  3. transparent load balancer – balance load and forward packets to the right machine in a cluster of two with same IP address
  4. HTTP protocol translator – add support in the transparent load balancer for HTTP streaming having an RTSP/RTP server

About INF5062: Exam

- **Prerequisite – mandatory assignments:**
  - lab assignment 1: protocol statistics
  - presentation of a relevant paper
- **Graded assignments (counting 33% each):**
  - lab assignment 3: transparent load balancer
    - deliver code
    - short demo/explanation of code
  - lab assignment 4: HTTP protocol translator
    - deliver code and a short report
    - present and demonstrate
- **Final oral exam (counting 33%):** early December 2006
  - excerpts IXP documentation
  - lecture slides
  - presented papers
  - content of lab assignments

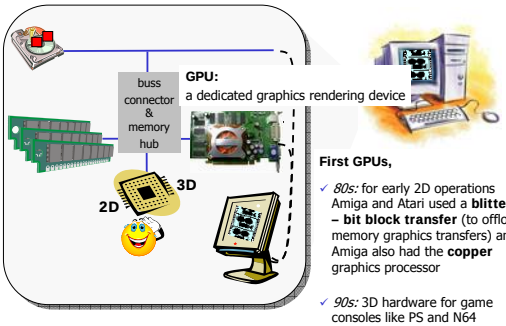
Available Resources

- Resources will be placed at
  - http://www.ifi.uio.no/~{paalh | griff}/INF5062
  - Login: *inf5062*
  - Password: *ixp*
  - Manuals, papers, code example, …

Background and Motivation 1:

Graphics Processing Units

Graphics Processing Units (GPUs)



First GPUs,

- *80s:* for early 2D operations Amiga and Atari used a **blitter – bit block transfer** (to offload memory graphics transfers) and Amiga also had the **copper** graphics processor
- *90s:* 3D hardware for game consoles like PS and N64

General Purpose Computing on GPU

- The
  - high arithmetic precision
  - extreme parallel nature
  - optimized, special-purpose instructions
  - available resources
  - …

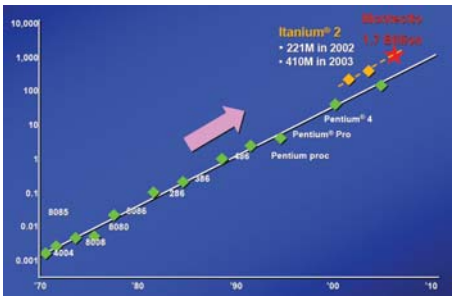
  … of the GPU allows for general, non-graphics related operations to be performed on the GPU
- **BUT: how should it be programmed and which tasks should go where?**

Background and Motivation 2:

Moore's Law for single cores

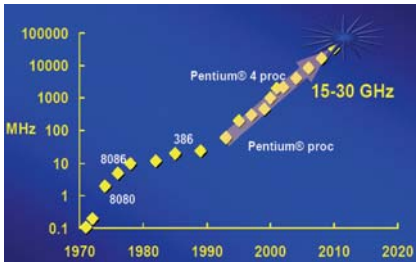
Motivation: Intel View

- ✓ Soon >billion transistors integrated



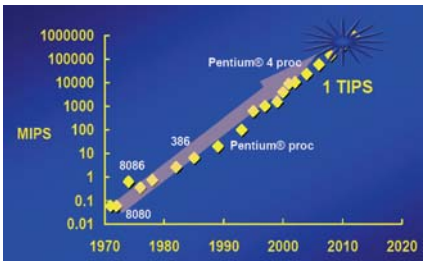
Motivation: Intel View

- ✓ Soon >billion transistors integrated
- ✓ Clock frequency **can** still increase

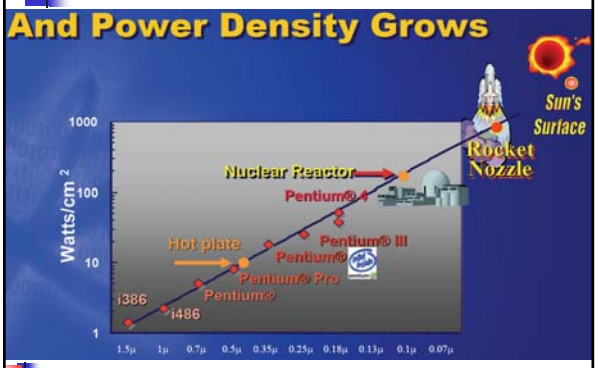


Motivation: Intel View

- ✓ Soon >billion transistors integrated
- ✓ Clock frequency **can** still increase
- ✓ Future applications will demand TIPS



Motivation: Intel View



Motivation

"Future applications will demand TIPS"

"Think platform beyond a single processor"

"Exploit concurrency at multiple levels"

"Power will be the limiter due to complexity and leakage"

Distributed workload on multiple cores
+ simple processors consume less energy

↳ **asymmetric multi-core processors**

Co-Processors

- ✓ Commodore Amiga was one of the earlier machines that used multiple processors
  - blitter & copper (as we saw for the GPUs)
  - Motorola 680x0
  - IBM power
  - old Motorola kept for backwards compatibility and parts of the OS not ported
- ✓ The original IBM PC included a socket for an Intel 8087 floating point co-processor (FPU)

  ➡ 50-fold speed up of floating point operations
- ✓ Intel kept the co-processor up to i486, where the i487 actually was a full 486DX knocking the 486SX to sleep.

Intel Multi-Core Processors

- ✓ Single-die multi-processors is the new era and the next logical step in driving Moore's Law into another decade

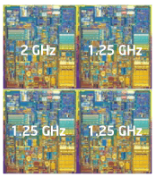
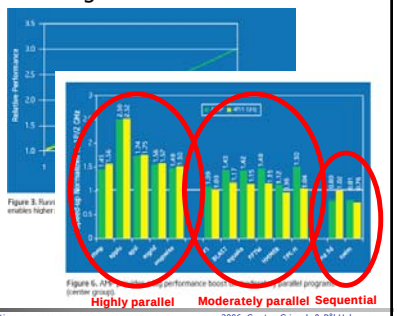


Figure 5. In an asymmetric multi-processor, the cores can run at different speeds to dynamically vary the EPI.

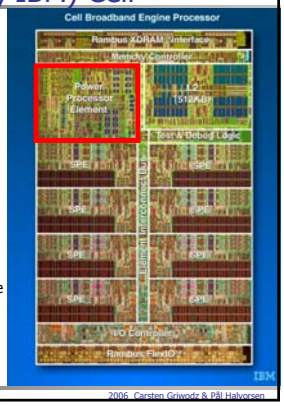


Intel Multi-Core Processors

- ✓ The operating systems can handle only a limited number of threads, e.g., 64 in Windows (2005)
- ✓ Where does the doubling stop?
  How many applications need more than 64 threads?
  Performance?
  ➡ software limits is the issue
- ✓ Application specific engines?
  ➡ Intel Academic Forum 2006: YES
  ➡ **But: Programming model?????**

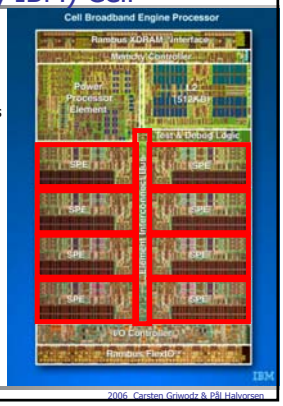
STI (Sony, Toshiba, IBM) Cell

- ✓ Cell is a 9-core processor
  - combining a light-weight general-purpose processor with multiple co-processors into a coordinated whole
  - *Power Processing Element* (PPE)
    - conventional Power processor
    - not supposed to perform all operations itself, acting like a controller
    - running conventional OSes
    - 16 KB instruction/data level 1 cache
    - 512 KB level 2 cache



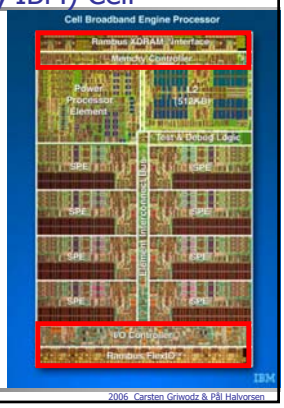
STI (Sony, Toshiba, IBM) Cell

- *Synergistic Processing Elements* (SPE)
  - specialized co-processors for specific types of code, i.e., very high performance vector processors
  - local stores
  - can do general purpose operations
  - the PPE can start, stop, interrupt and schedule processes running on an SPE
- Element Interconnect Bus (EIB)
  - internal communication bus
  - connects on-chip system elements:
    - PPE & SPEs
    - the memory controller (MIC)
    - two off-chip I/O interfaces
  - 25.6 GBps each way



STI (Sony, Toshiba, IBM) Cell

- memory controller
  - Rambus XDRAM interface to Rambus XDR memory
  - dual channels at 12.8 GBps → 25.6 GBps
- I/O controller
  - Rambus FlexIO interface which can be clocked independently
  - dual configurable channels
  - maximum ~ 76.8 GBps



STI (Sony, Toshiba, IBM) Cell

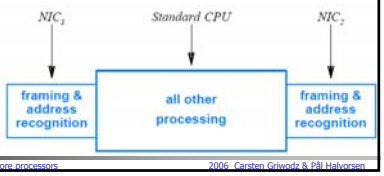
- Cell has in essence traded running everything at moderate speed for the ability to run certain types of code at high speed
- used for example in
  - Sony PlayStation 3:
    - 3.2 GHz clock
    - 7 SPEs for general operations
    - 1 SPE for security for the OS
  - Toshiba home cinema:
    - decoding of 48 HDTV MPEG streams
      → dozens of thumbnail videos simultaneously on screen
  - IBM blade centers:
    - 3.2 GHz clock
    - Linux ≥ 2.6.11

Background and Motivation 3:

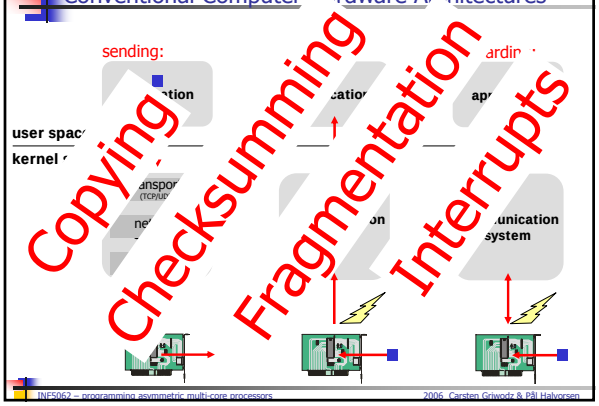
Network traffic increase

Software-Based Network System

- ✓ Uses conventional, shared hardware (e.g., a PC)
- ✓ Software
  - runs the entire system
  - allocates memory
  - controls I/O devices
  - performs all protocol processing
- ✓ First generation network systems:



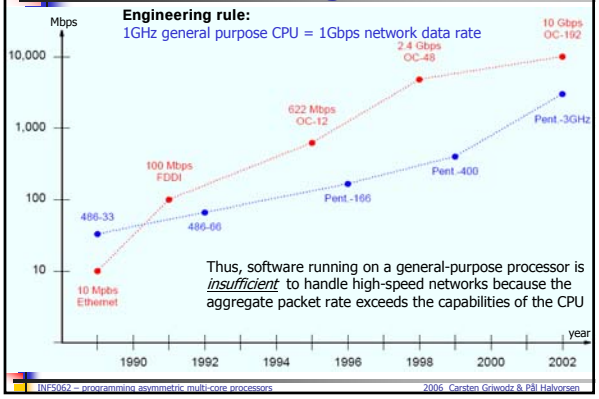
Review of General Data Path on Conventional Computer Hardware Architectures



Question:

- ✓ Which is growing faster?
  - network bandwidth
  - processing power
- ✓ Note: if network bandwidth is growing faster
  - CPU may be the bottleneck
  - need special-purpose hardware
  - conventional hardware will become irrelevant
- ✓ Note: if processing power is growing faster
  - no problems with processing
  - network/busses will be bottlenecks

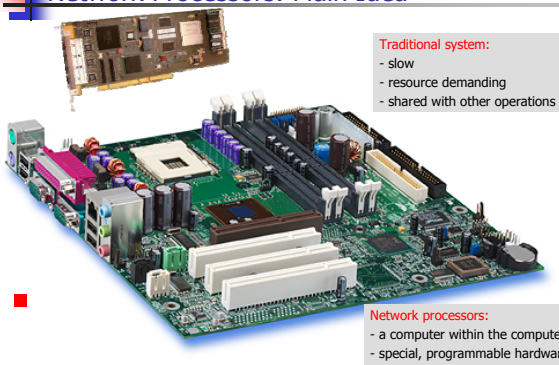
Growth Of Technologies



Network Processors: The Idea in a Nutshell

- ✓ Many designs through many generations (varying amount of HW & SW)
- ✓ Include support for protocol processing and I/O on one chip
  - General-purpose processor(s) for control tasks
  - Special-purpose processor(s) for packet processing and table lookup
  - Include functional units for tasks such as checksum computation, hashing, …
- ⇨ Call the result a ***network processor***
- ✓ We are here – they exist
- ⇨ **BUT: programming model??**

Network Processors: Main Idea



Traditional system:
- slow
- resource demanding
- shared with other operations

Network processors:
- a computer within the computer
- special, programmable hardware
- offloads host resources

Explosion of Commercial Products

- 1990 → 2000: network processors transformed from interesting curiosity to mainstream product
  - reduction in both overall costs and time to market
  - 2002: over 30 vendors with a vide range of architectures
  - e.g.,
    - Multi-Chip Pipeline (Agere)
    - Augmented RISC Processor (Alchemy)
    - Embedded Processor Plus Coprocessors (Applied Micro Circuit Corporation)
    - Pipeline of Homogeneous Processors (Cisco)
    - Pipeline of Heterogeneous Processors (EZchip)
    - Configurable Instruction Set Processors (Cognigine)
    - Extensive And Diverse Processors (IBM)
    - Flexible RISC Plus Coprocessors (Motorola)
    - Internet Exchange Processor (Intel)
    - …

Agere PayloadPlus: A Short Overview

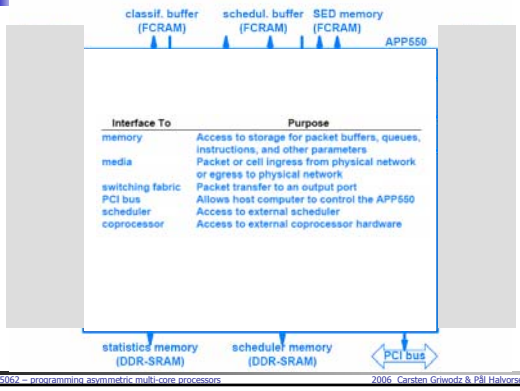
Agere PayloadPlus (APP)

- Agere PayloadPlus (APP)
  - consists of both programmable hardware and software
  - consists of both data and control planes (i.e., slow and fast plane)
- APP defines HW architectures, SW mechanisms, interconnection mechanisms and interfaces, **BUT** does not specify how to implement them
- Several versions of APP exist differing in the number and types of functional units, degree of parallelism and internal bandwidth (2. generation: 5 models)

APP Conceptual Pipeline

- *Classifier*
  - extract packets from ingress
  - classify packet
  - send statistics to state engine
  - reassemble blocks
  - pass packet to forwarder together with classification decision
- *State engine*
  - initiate, configure and control classifier and traffic manager
  - receives control from classifier
  - update statistics (e.g., packet count)
  - check packets against profiles (and inform classifier)
- *Forwarder*
  - get packet from classifier
  - perform traffic shaping and management
  - fragment packet (if necessary)
  - modify headers (if necessary)

APP550 Chip



Interface To	Purpose
memory	Access to storage for packet buffers, queues, instructions, and other parameters
media	Packet or cell ingress from physical network or egress to physical network
switching fabric	Packet transfer to an output port
PCI bus	Allows host computer to control the APP550
scheduler	Access to external scheduler
coprocessor	Access to external coprocessor hardware

APP550 Chip

classif. buffer (FCRAM) | schedul. buffer (FCRAM) | SED memory (FCRAM) | APP550

Memory interfaces:
- two types of physical memory
  - fast cycle RAM (FCRAM) for fast memory accesses
  - double data rate SRAM (DDR-SRAM) for high throughput
- the different memory types are usually used like this:

Engine	Memory	Use
Classifier	FCRAM	Packet buffer memory
	FCRAM	Program memory (patterns)
	FCRAM	Control memory
State Engine	DDR-SRAM	Flow statistics and profile memory
	DDR-SRAM	OAM data memory
Stream Editor	DDR-SRAM	Context memory
	FCRAM	Parameter memory
Reorder Buffer and Shaper	FCRAM	Scheduler buffer memory
	DDR-SRAM	Scheduler linked list memory
	DDR-SRAM	Scheduler parameter memory
	DDR-SRAM	Scheduler queue memory

statistics memory (DDR-SRAM) | scheduler memory (DDR-SRAM) | PCI bus

APP550 Chip

classif. buffer (FCRAM) | schedul. buffer (FCRAM) | SED memory (FCRAM) | APP550

Media interfaces:
- several to form fast data paths
- two external connections:
  - cell-oriented (ATM)
  - packet-oriented (Ethernet)

Standard Name	Meaning
GMII	Gigabit Media Independent Interface
PMII	Physical Media Independent Interface
SMII	Serial Media Independent Interface
PMA	Physical Medium Attachment
SPI-3	System Packet Interface Level 3
Utopia	A well-known data transfer mechanism

statistics memory (DDR-SRAM) | scheduler memory (DDR-SRAM) | PCI bus

APP550 Chip

classif. buffer (FCRAM) | schedul. buffer (FCRAM) | SED memory (FCRAM) | APP550

Scheduling interface interfaces:
- an external scheduling interface
- external logic can use information about queues

PCI bus interfaces:
- allows communication with host CPU
- mainly to control the whole operation

statistics memory (DDR-SRAM) | scheduler memory (DDR-SRAM) | PCI bus

APP550 Chip

classif. buffer (FCRAM) | schedul. buffer (FCRAM) | SED memory (FCRAM) | APP550

Coprocessor interfaces:
- APP550 should be able to process a packet
- BUT, to accommodate special cases, e.g., adding additional headers a co-processor interface is provided

statistics memory (DDR-SRAM) | scheduler memory (DDR-SRAM) | PCI bus

APP550 Chip

APP550 Chip

Stream Editor (SED)
- two parallel engines
- modify outgoing packets (e.g., checksum, TTL, ...)
- configurable, but not programmable

Packet (protocol data unit) assembler
- collect all blocks of a frame
- not programmable

Pattern Processing Engine
- patterns specified by programmer
- programmable using a special high-level language
- only pattern matching instructions
- parallelism by hardware using multiple copies and several sets of variables
- access to different memories

PDU assembler

Reorder Buffer Manager
- transfers data between classifier and traffic manager
- ensure packet order due to parallelism and variable processing time in the pattern processing

output interface

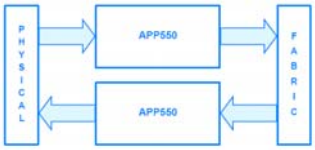
pattern processing engine | reorder buffer and traffic shaper | state engine

Traffic Manager
- schedule packets and shape traffic flow
- programmable via scripts
- sends packets to output interface
- according to implemented policy:
  - discard packets
  - choose queue

State Engine
- gather information (statistics) for scheduling
- verify flow within bounds
- provide an interface to the host
- configure and control other functional units

APP550 Full Duplex

- ✓ Clock rate for APP550 is 233 MHz
- ✓ One chip cannot manage packet at wire speed in both directions – often two in parallel (one each direction)



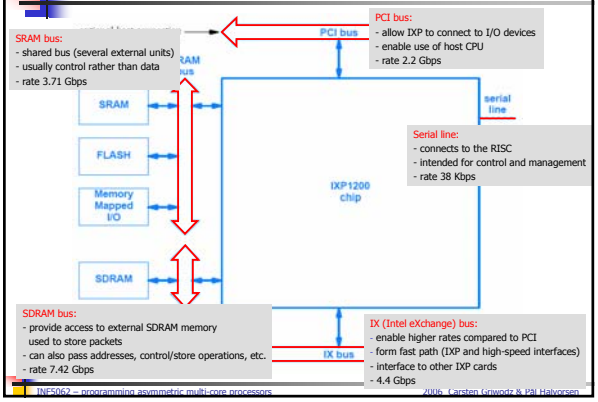
- overhead: all features needed in both direction?
- classification only one direction
  → checks outgoing packets and enqueues using special queue

Intel IXP1200 / 2400: A Short Overview

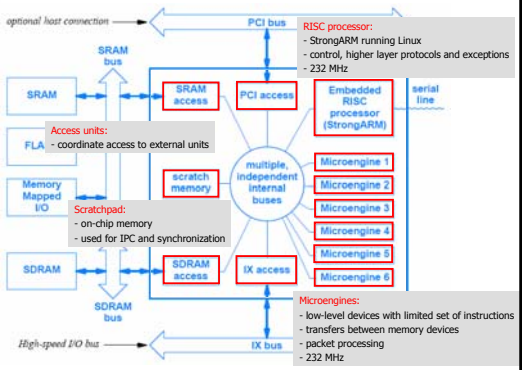
IXA: Internet Exchange Architecture

- ✓ IXA is a broad term to describe the Intel network architecture (HW & SW, control- & data plane)
- ✓ IXP: Internet Exchange Processor
  - processor that implements IXA
  - IXP1200 is the first IXP chip (4 versions)
  - IXP2xxx has now replaced the first version
- ✓ **IXP1200** basic features
  - 1 embedded 232 MHz StrongARM
  - 6 packet 232 MHz *µengines*
  - onboard memory
  - 4 x 100 Mbps Ethernet ports
  - multiple, independent busses
  - low-speed serial interface
  - interfaces for external memory and I/O busses
  - …
- ✓ **IXP2400** basic features
  - 1 embedded 600 MHz XScale
  - 8 packet 600 MHz *µengines*
  - 3 x 1 Gbps Ethernet ports
  - …

IXP1200 Architecture



IXP1200 Architecture



IXP1200 Processor Hierarchy

Processor Type	Onboard?	Programmable?

General-Purpose Processor:
- used for control and management
- running general applications

RISC processor:
- chip configuration interface (serial line)
- control, higher layer protocols and exceptions

I/O processors (microengines):
- transfers between memory devices
- packet processing

Coprocessors:
- real-time clock and timers
- IX bus controller
- hashing unit
- …

Physical interface processors:
- implement layer 1 & 2 processing

IXP1200 Memory Hierarchy

Memory Type	Maximum Size	On Chip?	Typical Use
GP Registers	128 regs.	yes	Intermediate computation
Inst. Cache	16 Kbytes	yes	Recently used instructions
Data Cache	8 Kbytes	yes	Recently used data
Mini Cache	512 bytes	yes	Data that is reused once
Write buffer	unspecified	yes	Write operation buffer
Scratchpad	4 Kbytes	yes	IPC and synchronization
Inst. Store	64 Kbytes	yes	Microengine instructions
FlashROM	8 Mbytes	no	Bootstrap
SRAM	8 Mbytes	no	Tables or packet headers
SDRAM	256 Mbytes	no	Packet storage

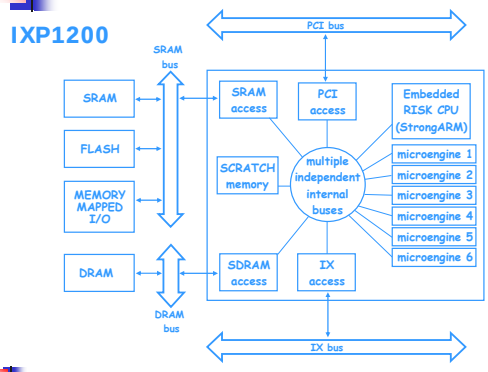
IXP1200 Memory Hierarchy

Memory Type	Addressable Data Unit (bytes)	Relative Access Time	Special Features

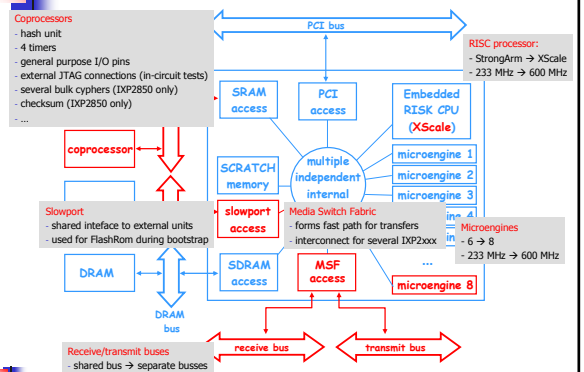
Different **memory** types…
- ✓…are organized into different addressable data units (words or longwords)
- ✓…have different access times
- ✓…connected to different busses

Therefore, to achieve optimal performance, programmers must ***understand the organization*** and ***allocate*** items from the ***appropriate type***

IXP1200 → IXP2400



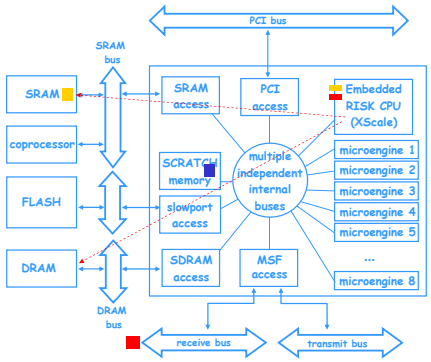
IXP2400 Architecture



IXP2400 Architecture

- ✓ Memory
  - generally more of everything
  - generally larger gap between CPUs and memory access in terms of cycles
  - **local memory** on each microengine
    - saving temporary results
    - private per packet processor
    - small (2560 bytes)
    - low latency (one cycle)
    - accessed through special registers

IXP2400 Basic Packet Processing



The End: Summary

- ✓ Asymmetric multi-core processors are already
- ⇨ **Challenge: programming**
  - **should know the capabilities of the system**
  - **identify which parts of a program that should run where**
  - **different methods to program the different components**
- ✓ We will use **Intel IXP2400** as an example which offers…
  - …embedded processor plus parallel packet processors
  - …connections to external memories and buses
- ✓ Next time: how to start programming these monsters