INF5062: Programming Asymmetric Multi-Core Processors

Memory

September 23, 08
Outline

- Traditional architectures
- IXP
- Cell
- nVidia

Know your architecture – memory types and access times are not uniform!!!
Traditional Architectures
Example:

Intel Hub Architecture (850 Chipset) – II

Pentium 4 Processor
- registers
- cache(s)

System bus
- (64-bit, 400/533 MHz → ~24-32 Gbps)

Memory controller hub

RAM interface
- (two 64-bit, 200 MHz → ~24 Gbps)

I/O controller hub

PCI bus
- (32-bit, 33 MHz → 1 Gbps)

Hub interface
- (four 8-bit, 66 MHz → 2 Gbps)

RDRAM
- RDRAM
- RDRAM

PCI slots
- PCI slots
- PCI slots
Example: AMD Opteron & Intel Xeon MP 4P servers

AMD Opteron™ Processor-based 4P Server

Intel Xeon MP Processor-based 4P Server

Key
- Memory Traffic
- I/O Traffic
- IPC Traffic
Example:
IBM POWER 4

POWER 4 chip

- CPU L1
- CPU L1
- core interface switch
- L2
- fabric controller (chip-chip fabric + multi-chip module)
- GX controller
- L3 controller

GX bus
(two 32-bit, 600 MHz → ~35 Gbps)

remote I/O (RIO) bridge

PCI busses
(32/64-bit, 33/66 MHz → 1-4 Gbps)

PCI host bridge

PCI-PCI bridge

PCI slots

RIO bus
(two 8-bit, 500 MHz → ~7 Gbps)

PCI host bridge

PCI-PCI bridge

PCI slots

L3 memory controller

RAM

PCI-PCI bridge

PCI slots

PCI host bridge

PCI-PCI bridge

PCI slots

RAM

RAM

RAM

(four 64-bit, 400 MHz → ~95 Gbps)

(four 64-bit, 400 MHz → ~95 Gbps)

(eight 32-bit, 400 MHz → ~95 Gbps)
**Example:** IBM POWER 4

Multichip modules in fabric controller can connect 4 chips into a 4 chip, 2-way SMP → 8-way MP
Memory access penalty of up to 10% in the multi-multichipmodule configuration.

Chip-chip fabric in fabric controller can connect 4 multi-chips into a 4x4 chip, 2-way SMP → 32-way MP.
IXP2400
IXP2400 Architecture

**IXP2400**

- **SRAM**
- **DRAM**
- **SRAM access**
- **DRAM access**
- **PCI access**
- **Embedded RISK CPU (XScale)**
  - Microengine 1
  - Microengine 2
  - Microengine 3
  - Microengine 4
  - Microengine 5
- **SCRATCH memory**
- **MSF access**
- **Multiple independent internal buses**
- **PCI bus**
- **Coprocessor**

---

*University of Oslo*

INF5062, Carsten Griwodz & Pål Halvorsen

[ simula.research.laboratory ]
**IXP2400 Architecture**

**IXP2400**

- **SRAM**
  - SRAM access
- **DRAM**
  - DRAM access
- **PCI bus**
- **Embedded RISK CPU (XScale)**
  - microengine 1
  - microengine 2
  - microengine 3
  - microengine 4
  - microengine 5
  - ... microengine 8

**Multiple independent internal buses**

- **SCRATCH memory**
- **PCI access**
- **MSF access**
Scratch Memory

- Recommended use
  - Passing messages between processors and between threads
  - Semaphores, mailboxes, other IPC

- 32-bit addressed (4 byte aligned, word aligned)
- 4 Kbytes
- Has an atomic auto-increment instruction
  - Only usable by microengines
Scratchpad Special Features

- Atomic bit set and clear with/without test
- Atomic increment/decrement
- Atomic add and swap
- Atomic get/put for rings
  - Hardware support for rings links SRAM
  - Signaling when ring is full
SRAM

- Recommended use
  - Lookup tables
  - Free buffer lists
  - Data buffer queue lists

- 32-bit addressed (4 byte aligned, word aligned)

- Up to 16 MB
  - Distributed over 4 channels
  - Our cards have 8 MB, use 2 channels

- 1.6 Gbps peak bandwidth
  - Lower bandwidth than SDRAM
  - Lower latency than SDRAM

Access
- XScale
- Microengines

Accessing SRAM
- XScale - Byte, word and longword access
- Microengine - Bit and longword access only
SRAM Special Features

- Atomic bit set and clear with/without test
- Atomic increment/decrement
- Atomic add and swap
- Atomic enqueue, enqueue_tail, dequeue
  - Hardware support for maintaining queues
  - Several modes
    - Queue mode: data structures at discontiguous addresses
    - Ring mode: data structures in a fixed-size array
    - Journaling mode: keep previous values in a fixed-size array
SDRAM

- **Recommended use**
  - XScale instruction code
  - Large data structures
  - Packets during processing

- **64-bit addressed (8 byte aligned, quadword aligned)**
- **Up to 2GB**
  - Our cards have 256 MB
  - Unused higher addresses map onto lower addresses!
- **2.4 Gbps peak bandwidth**
  - Higher bandwidth than SRAM
  - Higher latency than SRAM

- **Access**
  - Instruction from external devices are queued and scheduled
  - Accessed by
    - XScale
    - Microengines
    - PCI
IXP2400 Architecture

IXP2400

- SRAM
- coprocessor
- DRAM

Multiple independent internal buses

- SRAM access
- PCI access
- SDRAM access
- MSF access
- Embedded RISK CPU (XScale)
  - microengine 1
  - microengine 2
  - microengine 3
  - microengine 4
  - microengine 5
  - ...
  - microengine 8

PCI bus

SRAM bus

DRAM bus
XScale Memory

A general purpose processor
- With MMU
  - 32 Kbytes instruction cache
    - Round robin replacement
  - 32 Kbytes data cache
    - Round robin replacement
    - Write-back cache, cache replacement on read, not on write
  - 2 Kbytes mini-cache for data that is used once and then discarded
- Instruction code stored in SDRAM
## XScale Memory Mapping

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Memory Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFF FFFF</td>
<td>PCI Memory</td>
</tr>
<tr>
<td>0xE000 0000</td>
<td>Other</td>
</tr>
<tr>
<td>0xDFFF FFFF</td>
<td>SRAM</td>
</tr>
<tr>
<td>0xC000 0000</td>
<td>SDRAM</td>
</tr>
<tr>
<td>0xBFFF FFFF</td>
<td></td>
</tr>
<tr>
<td>0x8000 0000</td>
<td></td>
</tr>
<tr>
<td>0x7FFF FFFF</td>
<td></td>
</tr>
<tr>
<td>0x0000 0000</td>
<td></td>
</tr>
</tbody>
</table>

- **PCI Memory** (up to ½GB)
- **Other** (up to ½GB)
- **SRAM** (up to 1GB)
- **SDRAM** (up to 2GB)

### Available on our cards:
- 256 MB SDRAM
- 64 MB SRAM
- 16 KB Scratch

**PCI controller CSRs**
- PCI config registers
- PCI Spec/IACK
- PCI CFG
- PCI I/O
- **XScale Local CSRs (32MB)**
- reserved
- **DRAM CSRs**
- **SRAM CSRs & Queue Array (64MB)**
- **Scratch (32MB)**
- **MSF**
- Flash ROM
- reserved
- **CAP-CSRs**
XScale Memory Mapping

- **0xFFFF FFFF**
  - PCI Memory (up to ½GB)
- **0xE000 0000**
  - reserved
- **0xB000 0000**
  - SDRAM (up to 2GB)
- **0x8000 0000**
  - Channel 1
  - same memory mapped four times
  - implicit features
  - no more than 128 kB per channel
- **0x0000 0000**
  - Channel 0
  - same memory mapped four times
  - implicit features
  - no more than 128 kB per channel

**SRAM CSR**
- 0xCC40 0100
- 0xCC00 0100
- 0xCE40 0000
- 0xCE00 0000

**SRAM CSR**
- 0xDFFF FFFF
- 0xBFFF FFFF
- 0x8000 0000
- 0x7FFF FFFF

**Add, Test and Add**
- 0x9C00 0000
- 0x8C00 0000

**Bit Clear, Bit Test & Clear**
- 0x9800 0000
- 0x8800 0000

**Bit Set, Bit Test & Set**
- 0x9400 0000
- 0x8400 0000

**Read, Write, Swap**
- 0x9000 0000
- 0x8000 0000

**Get, Put**
- 0xCE01 0000
- 0xCE41 0000

**Deq, Enq**
- 0xCC00 0100
- 0xCC40 0100

**Deq, Enq**
- 0xCC01 0000
- 0xCC41 0000

**Get, Put**
- 0xCE00 0000
- 0xCE40 0000
IXP2400 Architecture

**IXP2400**

- SRAM
- coprocessor
- DRAM

**Buses and Connections**

- SRAM bus
- DRAM bus
- PCI bus
- Multiple independent internal buses
- Embedded RISK CPU (XScale)

**Microengine Clusters**

- Microengine cluster 1
  - microengine 1
  - microengine 2
  - microengine 3

- Microengine cluster 2
  - microengine 4
  - microengine 5
  - microengine 6
  - microengine 7
  - microengine 8
Microengine Memory

- 256 general purpose registers
  - Arranged in two banks

- 512 transfer registers
  - Transfer registers are not general purpose registers
  - DRAM transfer registers
    - Transfer in
    - Transfer out
  - SRAM transfer registers
    - Transfer in
    - Transfer out
  - Push and pull on transfer registers usually by external units

- 128 next neighbor registers
  - New in ME V2
  - Dedicated data path to neighboring ME
  - Also usable inside a ME
  - SDK use: message forwarding using rings

- 2560 bytes local memory
  - New in ME V2
  - RAM
  - Quad-aligned
  - Shared by all contexts
  - SDK use: register spill in code generated from MicroC
These are the header fields you need for the web bumper:

- Ethernet type = \textbf{0x800}
- IP type = \textbf{6}
- TCP port = \textbf{80}
### Data Flow & Operations

#### Network Packet Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dest</td>
<td>Destination Address</td>
</tr>
<tr>
<td>src (0-1)</td>
<td>Source Address</td>
</tr>
<tr>
<td>src (2-4)</td>
<td>Source Address</td>
</tr>
<tr>
<td>type</td>
<td>Type</td>
</tr>
<tr>
<td>vers</td>
<td>Version</td>
</tr>
<tr>
<td>hlen</td>
<td>Header Length</td>
</tr>
<tr>
<td>service</td>
<td>Service</td>
</tr>
<tr>
<td>total length</td>
<td>Total Length</td>
</tr>
<tr>
<td>identification</td>
<td>Identification</td>
</tr>
<tr>
<td>flgs</td>
<td>Flags</td>
</tr>
<tr>
<td>frag. offset</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>ttl</td>
<td>Time to Live</td>
</tr>
<tr>
<td>type</td>
<td>Type</td>
</tr>
<tr>
<td>header checksum</td>
<td>Header Checksum</td>
</tr>
<tr>
<td>src. address</td>
<td>Source Address</td>
</tr>
<tr>
<td>dest. address (0-1)</td>
<td>Destination Address</td>
</tr>
<tr>
<td>dest. address (2-3)</td>
<td>Destination Address</td>
</tr>
<tr>
<td>src. port</td>
<td>Source Port</td>
</tr>
<tr>
<td>dest. port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>seq. number (0-1)</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>seq. number (2-3)</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>acknowledgement</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>vers</td>
<td>Version</td>
</tr>
<tr>
<td>reserv.</td>
<td>Reserved</td>
</tr>
<tr>
<td>code</td>
<td>Code</td>
</tr>
<tr>
<td>window</td>
<td>Window</td>
</tr>
<tr>
<td>checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>urgent pointer</td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>data</td>
<td>Data</td>
</tr>
<tr>
<td>total length</td>
<td>Total Length</td>
</tr>
<tr>
<td>identification</td>
<td>Identification</td>
</tr>
<tr>
<td>flgs</td>
<td>Flags</td>
</tr>
<tr>
<td>frag. offset</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>ttl</td>
<td>Time to Live</td>
</tr>
<tr>
<td>type</td>
<td>Type</td>
</tr>
<tr>
<td>header checksum</td>
<td>Header Checksum</td>
</tr>
<tr>
<td>src. address</td>
<td>Source Address</td>
</tr>
<tr>
<td>dest. address (0-1)</td>
<td>Destination Address</td>
</tr>
<tr>
<td>dest. address (2-3)</td>
<td>Destination Address</td>
</tr>
<tr>
<td>src. port</td>
<td>Source Port</td>
</tr>
<tr>
<td>dest. port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>seq. number (0-1)</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>seq. number (2-3)</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>acknowledgement</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>vers</td>
<td>Version</td>
</tr>
<tr>
<td>reserv.</td>
<td>Reserved</td>
</tr>
<tr>
<td>code</td>
<td>Code</td>
</tr>
</tbody>
</table>

#### SDRAM Memory System

- **SDRAM Unit**: Contains transfer registers for memory operations.
- **RFIFO** and **TFIFO**: FIFO buffers for data transfer.
- **SDRAM Read Bus**: Connects SDRAM unit to other components.
- **wwbump**: A control signal for SDRAM operations.

#### Microengine

- **Transfer Registers**: General-purpose registers for data transfer.
- **SDRAM Read Transfer Registers**: Specific registers for SDRAM read operations.
Cell Broadband Engine
Cell Broadband Engine Structure
Cell Broadband Engine Structure

PPE - Power Processing Element
SPE - Synergistic Processing Element
MIC - Memory Interface Controller
EIB - Element Interconnect Bus
- 4 rings
- Each 128 bytes wide (128 bytes concurrently in transfer)
- Separate send and receive “ramps”
- **MIC**
  - One or two RAMBUS XDR data interfaces
  - Read or write lengths: 1, 2, ..., 8, 16, 32, 64 or 128 bytes
  - 64 entry read queue, 64 entry write queue
  - Up to 64 GB
Cell Broadband Engine Structure

- **PPE**
  - Power processing element

- **PPU**
  - Power processing unit
  - 32 kB L1 instruction cache
  - 32 kB L2 data cache

- **PPSS**
  - PowerPC processor storage subsystem
  - 512 kB unified L2 cache
  - EIB accesses are sequential and ordered

In the diagram:
- **PPE**
- **PPU**
- **PPSS**
- L1 instruction cache
- L1 data cache
- L2 cache

EIB accesses are sequential and ordered.
Cell Broadband Engine Structure

- **PPU**
  - 2-way symmetric multithreading

- **Duplicated register set**
  - 32 64-bit general purpose registers
  - 32 64-bit floating point registers
  - 32 128-bit vector registers
  - 1 link register for branching
  - 1 count register for loop count or branching
Cell Broadband Engine Structure

- **SPE**
  - Synergetic processing element

- **SPU**
  - Synergetic processing unit
  - 4 execution units (for 4 hardware contexts)
  - RF: register file
  - LS: local store

- **MFC: Memory flow controller**
  - MMIO registers: memory mapped I/O registers
  - DMA controller
Cell Broadband Engine Structure

- **Register file**
  - 128 registers
  - Each 128 bits wide

- **Local store**
  - 256 KB
  - 128 bit-aligned access
  - transparent (but less efficient) read-modify-write cycle for unaligned access
  - Access times and priority
    1. MMIO 16 bytes/cycle
    2. DMA 128 bytes/cycle
    3. Data fetch 16 bytes/cycle
    4. Instr. fetch 128 bytes/cycle
Cell Broadband Engine Structure

- **DMA controller**
  - Can have 16 DMA operations concurrently in flight, more than normal CPUs (i.e. 128 total)
  - Transfer size up to 16 KB for each operation

- **Mailboxes**
  - 32-bit dedicated registers
  - Implemented as channels
  - Used for communication with PPE
  - 2 outgoing mailboxes
  - 1 incoming mailbox
Cell Broadband Engine Structure

- **SPE**
  - Each SPE’s memory controller can have 16 DMA operations concurrently in flight, more than normal CPUs (i.e. 128 total)

- **Observation**
  - SPEs can manage memory accesses without PPE help
Cell Broadband Engine Structure

- **Either**
  - 64 bit flat address space
  - No mapping of local store to PPE address space

- **Or**
  - Virtual addressing
  - 42 bit real address space
    - Independently set up for data and instruction addressing
  - 64 bit effective address space
  - 65 bit virtual address space

- 256 MB segment sizes
  - Mappable from 65-bit space to 42-bit space

- 4kB and 2 of (16kB, 1MB, 16MB) page sizes
  - Swappable

Flat address space of the PPE
Main storage (effective address space)

DMA requests can be sent to an MFC either by software on its associated SPU or on the PPE, or by any other processing device that has access to the MFC's MMIO problem-state registers.
nVIDIA
Kinds of memory

- Constant memory cache, texture memory cache, local memory, shared memory and registers are on chip.

- Texture memory, constant memory and global memory are in device memory.

- Reads from texture memory and constant memory are cached.

- Reads from and writes to local and global memory are not cached.
Kinds of memory

- Coherency mechanisms: no such thing

- Global, shared
  - no, explicit synchronization

- Constant, texture
  - is read-only, 4 cycles per read, broadcast capable

- Local
  - isn’t shared

- Global
  - not cached
Threads per SM

- **L1 instr. cache**
- **Shared memory**
- **8192 registers/SM**
- **64 KB Constant memory**
- **8 KB Texture memory**
- **8 KB Constant mem cache**
- **8 KB Texture mem cache**
- **16 KB Shared memory**
Multiprocessor facts

- The maximum number of threads per block is 512
- The maximum sizes of the x-, y-, and z-dimension of a thread block are 512, 512, and 64, respectively
- The maximum size of each dimension of a grid of thread blocks is 65535
- The warp size is 32 threads
- The maximum number of active blocks per multiprocessor is 8
- The maximum number of active warps per multiprocessor is 24
- The maximum number of active threads per multiprocessor is 768
Threads per SM

- accessing a register is zero extra clock cycles per instruction
  - delays may occur due to register read-after-write dependencies and register memory bank conflicts

- delays introduced by read-after-write dependencies can be ignored as soon as there are at least 192 active threads per multiprocessor to hide them

- thread scheduler works best when the number of threads per block is a multiple of 64
Performance

- Speed Host - device memory
  - PCIe x16: 3.2 GB/s

- Speed device memory - shared memory
  - 80 GB/s

- Counted by cycles

- Equivalent to clock speed of the card
  - 1.4 GHz on the GeForce GTX 280 GPU
Coalescing

- fast, bursty transfer between multiple registers allocated to different threads in the same warp
- if per-thread memory accesses for a single warp (today’s hardware actual half-warp) form a contiguous range of addresses, accesses will be coalesced into a single access

- is *not a dynamic ability*
- compiler must identify it and generate appropriate code

- occurs when thread \( n \) accessed "base-address + size * \( n \)”, where
  - base-address is aligned to 16 * size
  - size of size of the read operation: 4, 8 or 16 bytes
Memory latency and features

- Read from global memory
  - 16-bit aligned, otherwise read-modify-write
  - 32-bit, 64-bit, 128-bit read operations supported

- Coalesced read
  - If all threads in a warp read concurrently in a pattern, reads are “coalesced”
  - coalesced read of 32 bits per thread is slighter faster than coalesced read of 64 bits per thread and much faster than coalesced read of 128 bits per thread
  - but all are warpsize times faster than non-coalesced reads
Memory access

- Access to device memory is slow
  - 4 cycles to set up, 400 to 600 cycles to complete

- Use multithreading to use that time

- Partition your computation to keep the GPU multiprocessors equally busy
  - Many threads, many thread blocks

- Keep resource usage low enough to support multiple active thread blocks per multiprocessor
  - Registers, shared memory
Shared memory hints

Shared memory

- amount available per multiprocessor is 16 KB
- organized into 16 banks

- 32 bits readable from each bank in 2 clock cycles
- concurrent access by several threads of a warp leads to a bank conflict
- avoid high-degree bank conflicts
Shared memory

- Hundreds of times faster than global memory
- Threads can cooperate via shared memory

- Use one / a few threads to load / compute data shared by all threads

- Use it to avoid non-coalesced access
  - Stage loads and stores in shared memory to re-order non-coalescence address
Bank use

Thread 0: Address 128
Thread 1: Address 132
Thread 2: Address 136
Thread 3: Address 140
Thread 4: Address 144
Thread 5: Address 148
Thread 6: Address 152
Thread 7: Address 156
Thread 8: Address 160
Thread 9: Address 164
Thread 10: Address 168
Thread 11: Address 172
Thread 12: Address 176
Thread 13: Address 180
Thread 14: Address 184
Thread 15: Address 188

GOOD

BAD
Bank use

Thread 0
Address 128
Thread 1
Address 132
Thread 2
Address 136
Thread 3
Address 140
Thread 4
Address 144
Thread 5
Address 148
Thread 6
Address 152
Thread 7
Address 160
Thread 8
Address 164
Thread 9
Address 168
Thread 10
Address 172
Thread 11
Address 176
BAD
Thread 2
Address 152
Thread 3
Address 164
Thread 4
Address 176
Thread 5
Address 188
Thread 0
Address 128
Thread 1
Address 132
Thread 2
Address 136
Thread 3
Address 140
Thread 4
Address 144
Thread 5
Address 148
Thread 6
Address 152
Thread 7
Address 156
Thread 8
Address 160
Thread 9
Address 164
Thread 10
Address 168
Thread 11
Address 172
Thread 12
Address 176
Thread 13
Address 180
Thread 14
Address 184
Thread 15
Address 188
BAD
Thread 6
Address 152
Thread 7
Address 156
Thread 8
Address 160
Thread 9
Address 164
Thread 10
Address 168
Thread 11
Address 172
Thread 12
Address 176
Thread 13
Address 180
Thread 14
Address 184
Thread 15
Address 188
Texture memory

- Texture memory
  - optimize for spatial locality to increase cache performance

- Features and advantages
  - Reads are cached, useful if accesses have locality
  - No coalescing or bank conflict problems
  - Address calculation latency is hidden
  - Packed data can be delivered to several registers in a single operation
  - 8-bit and 16-bit integer data may be converted to 32-bit floating on the fly
Optimization priorities

- Memory coalescing has highest priority
  - Largest effect
  - Optimize for locality

- Take advantage of shared memory
  - Very high bandwidth
  - Threads can cooperate to save work

- Use parallelism efficiently
  - Keep the GPU busy at all times
  - High arithmetic / bandwidth ratio
  - Many threads & thread blocks

- Leave bank conflicts for last
  - 4-way and smaller conflicts are not usually worth avoiding if avoiding them will cost more instructions