Lecture 17 - Stream Processors (Part I)
Outline

• Hardware strengths and the stream execution model
• Stream Processor hardware
  – Parallelism
  – Locality
  – Hierarchical control and scheduling
  – Throughput oriented I/O
• Implications on the software system
  – Current status
• HW and SW tradeoffs and tuning options
  – Locality, parallelism, and scheduling
• Petascale implications
Stream Processors Offer Efficiency and Performance

Huge potential impact on petaflop system design
**Hardware Efficiency → Greater Software Responsibility**

- **Hardware matches VLSI strengths**
  - Throughput-oriented design
  - Parallelism, locality, and partitioning
  - Hierarchical control to simplify instruction sequencing
  - Minimalistic HW scheduling and allocation

- **Software given more explicit control**
  - Explicit hierarchical scheduling and latency hiding (**schedule**)
  - Explicit parallelism (**parallelize**)
  - Explicit locality management (**localize**)

---

**Must reduce HW “waste” but no free lunch**
Outline

• Hardware strengths and the stream execution model

• Stream Processor hardware
  – Parallelism
  – Locality
  – Hierarchical control and scheduling
  – Throughput oriented I/O

• Implications on the software system
  – Current status

• HW and SW tradeoffs and tuning options
  – Locality, parallelism, and scheduling

• Petascale implications
Effective Performance on Modern VLSI

• Parallelism
  - 10s of FPUs per chip
  - Efficient control

• Locality
  - Reuse reduces global BW
  - Locality lowers power

• Bandwidth management
  - Maximize pin utilization
  - Throughput oriented I/O (latency tolerant)

Parallelism, locality, bandwidth, and efficient control (and latency hiding)
## Bandwidth Dominates Energy Consumption

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy (0.13um)</th>
<th>Energy (0.05um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32b ALU Operation</td>
<td>5pJ</td>
<td>0.3pJ</td>
</tr>
<tr>
<td>Read 32b from 8KB RAM</td>
<td>50pJ</td>
<td>3pJ</td>
</tr>
<tr>
<td>Transfer 32b across chip (10mm)</td>
<td>100pJ</td>
<td>17pJ</td>
</tr>
<tr>
<td>Transfer 32b off chip (2.5G CML)</td>
<td>1.3nj</td>
<td>400pJ</td>
</tr>
</tbody>
</table>

1:20:260 local to global to off-chip ratio yesterday
1:56:1300 tomorrow

Off-chip >> global >> local >> compute
Stream Execution Model Accounts for Infinite Data
Stream Execution Model Accounts for Infinite Data
Stream Execution Model Accounts for Infinite Data

Process streams of “bite-sized” data (predetermined sequence)
Generalizing the Stream Model

• Data access determinable well in advance of data use
  – Latency hiding
  – Blocking

• Reformulate to **gather - compute - scatter**
  – Block phases into **bulk operations**

• “Well in advance”: enough to hide latency between blocks and SWP

• Assume data parallelism within compute phase
Take Advantage of Software: Hierarchical Bulk Operations

- Data access determinable **well in advance** of data use
  - Latency hiding
  - Blocking

- Reformulate to **gather - compute - scatter**
  - Block phases into **bulk operations**
Bulk Operations are Good for Hardware

- **Parallelism**
  - 10s of FPUs per chip
  - Efficient control
- **Streaming style**
  - explicit
  - positively constrained
  - scalable DLP
    - DLP >> SIMD
  - “memory level parallelism”

Parallelism is explicit and compact
Control is hierarchical and efficient
Bulk Operations are Good for Hardware

- **Parallelism**
  - 10s of FPUs per chip
  - Efficient control

- **Locality**
  - Reuse reduces global BW
  - Locality lowers power

- **Streaming style**
  - Explicit locality
  - Positively constrained
Bulk Operations are Good for Hardware

- **Parallelism**
  - 10s of FPUs per chip
  - Efficient control

- **Locality**
  - Reuse reduces global BW
  - Locality lowers power

- **Streaming style**
  - explicit locality
  - positively constrained
  - explicit communication

**Locality is explicit and compact**

**Communication is explicit**
Bulk Operations are Good for Hardware

- **Parallelism**
  - 10s of FPUs per chip
  - Efficient control

- **Locality**
  - Reuse reduces global BW
  - Locality lowers power

- **Latency Tolerance**
  - Throughput oriented I/O
  - Increasing on-/off-chip latencies

- **Minimum control overhead**

---

Hardware designed for throughput and not latency (memory BW, FLOPS, bulk exceptions, bulk coherency, …)
Generalizing the Stream Model

- Medium granularity bulk operations
  - Kernels and stream-LD/ST
- Predictable sequence (of bulk operations)
  - Latency hiding, explicit communication
- Hierarchical control
  - Inter- and intra-bulk
- Throughput-oriented design
- Locality and parallelism
  - Kernel locality + producer-consumer reuse
  - Parallelism within kernels

Generalized stream model matches VLSI requirements
Outline

• Hardware strengths and the stream execution model

• Stream Processor hardware
  – Parallelism
  – Locality
  – Hierarchical control and scheduling
  – Throughput oriented I/O

• Implications on the software system
  – Current status

• HW and SW tradeoffs and tuning options
  – Locality, parallelism, and scheduling

• Petascale implications
Parallelism and Locality in Streaming Scientific Applications

**VLSI**

- **Parallelism**
  - 10s of FPUs per chip
  - Efficient control

- **Locality**
  - Reuse reduces global BW
  - Locality lowers power

- **Bandwidth management**
  - Maximize pin utilization
  - Throughput oriented I/O (latency tolerant)

**Streaming model**

- medium granularity bulk operations
  - kernels and stream-LD/ST

- **Predictable sequence**

- **Locality and parallelism**
  - kernel locality + producer-consumer reuse
Stream Processor Architecture Overview

- **Parallelism**
  - Lots of FPUs
  - Latency hiding

- **Locality**
  - Partitioning and hierarchy

- **Bandwidth management**
  - Exposed communication (at multiple levels)
  - Throughput-oriented design

- **Explicit support of stream execution model**
  - Bulk kernels and stream load/stores

Maximize efficiency:

FLOPs / BW, FLOPs / power, and FLOPs / area
Stream Processor Architecture (Merrimac)

Multiple FPUs for high-performance

- 64-bit MADDs

64-bit MADDs

Multiple FPUs for high-performance
Stream Processor Architecture (Merrimac)

Need to bridge 100X bandwidth gap
Reuse data on chip and build locality hierarchy
Stream Processor Architecture (Merrimac)

LRF provides the bandwidth through locality
Low energy by traversing short wires
Stream Processor Architecture (Merrimac)

Clustering exploits kernel locality (short term reuse)

Cluster switch

64 64-bit MADDs (16 clusters)
Stream Processor Architecture (Merrimac)

Clustering exploits kernel locality (short term reuse) Enables efficient instruction-supply

- Cluster switch
- 64 64-bit MADDs (16 clusters)
- 64 GB/s
- <64 GB/s
- 3,840 GB/s
- ~61 KB
- 8 GB
- DRAM bank

Cluster switch
Stream Processor Architecture (Merrimac)

SRF reduces off-chip BW requirements (producer-consumer locality); enables latency-tolerance
Stream Processor Architecture (Merrimac)

Inter-cluster switch adds flexibility:
breaks strict SIMD and assists memory alignment
Stream Processor Architecture (Merrimac)

Cache is a BW amplifier for select accesses
Stream Processors

- ClearSpeed CSX600, MorphoSys, …
- GPUs?

Somewhat specialized processors but over a range of applications
Outline

• Hardware strengths and the stream execution model

• Stream Processor hardware
  – Parallelism
  – Locality
  – Hierarchical control and scheduling
  – Throughput oriented I/O

• Implications on the software system
  – Current status

• HW and SW tradeoffs and tuning options
  – Locality, parallelism, and scheduling

• Petascale implications
SRF Decouples Execution from Memory

Decoupling enables efficient static architecture: Separate address spaces (MEM/ SRF/ LRF)
Hierarchical Control

“Scalar” operations

Bulk memory operations

Bulk kernel operations

GPP Core

Staging area for bulk operations enables software latency hiding and high-throughput I/O
Hierarchical Control

Decoupling allows efficient and effective instruction schedulers
Streaming Memory Systems


DRAM systems are very sensitive to access pattern, Throughput-oriented memory system helps
Streaming Memory Systems Help

Capable memory system even more important for applications
Streaming Memory Systems

- Bulk stream loads and stores
  - Hierarchical control
- Expressive and effective addressing modes
  - Can’t afford to waste memory bandwidth
  - Use hardware when performance is non-deterministic

- Automatic SIMD alignment
  - Makes SIMD trivial (SIMD ≠ short-vector)

Stream memory system helps the programmer and maximizes I/O throughput