Stream Processors Offer Efficiency and Performance

Hardware Efficiency → Greater Software Responsibility

Stream Processors and GPUs

Effective Performance on Modern VLSI

Outline

Stream Processors

- Bulk kernel computation
  - Kernel uses “scalar” ISA
  - VLIW + SIMD
- Bulk memory operations
  - Software latency hiding
  - Stream mem. System
- HW optimized local mem.
  - Locality opportunities
- Minimize off-chip transfers
  - With capable mem system
- So far mostly load-time parameters

GPUs

- Bulk kernel computation
  - Kernel uses “scalar” ISA
  - SIMD
- Scalar mem. Operations
  - Threads to hide latency
  - Threads to fill mem. Pipe
- Small shared memory
  - Limited locality
- Rely on off-chip BW
  - Need for graphics
- Dynamic workloads
  - Mostly read-only

• 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
  - More details on HW and SW tradeoffs
  - Locality, parallelism, and scheduling
  - Irregular streaming applications

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>65nm</th>
<th>32nm</th>
<th>16nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>64b FP Operation</td>
<td>38</td>
<td>12.5</td>
<td>4.2</td>
</tr>
<tr>
<td>Read 64b from 1MB Cache</td>
<td>175</td>
<td>5.3</td>
<td>2</td>
</tr>
<tr>
<td>Transfer 64b across chip (10mm, Rep.)</td>
<td>175</td>
<td>179</td>
<td>175</td>
</tr>
<tr>
<td>Transfer 64b across chip (10mm, Cach)</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Transfer 64b off chip</td>
<td>154</td>
<td>115</td>
<td>100</td>
</tr>
</tbody>
</table>

Locality/Communication are key; Even then, performance is power-bound

Processor-Centric View - All Data Accessible

Realistic View - Only Part of Working Set is Accessible in On-Chip State

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
- "Well in advance": enough to hide latency between blocks and SWP
- Assume data parallelism within compute phase

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

Locality is explicit and compact
Communication is explicit
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
  - Memory issues
  - More details on HW and SW trade-offs
- Irregular streaming applications

Parallelism and Locality in Streaming Scientific Applications

VLSI
- Parallelism
  - Streaming chip vs. instruction level parallelism
- Locality
  - Granularity
  - Memory hierarchy
- Bandwidth management
  - Memory bandwidth
  - 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
- 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

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

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

Inter-cluster switch adds flexibility:
breaks strict SIMD and assists memory alignment

Cache is a BW amplifier for select accesses
Stream Processors

- ClearSpeed CSX600, MorphoSys, ...
- GPUs?

Somewhat specialized processors; very efficient over a range of high-performance applications

Stream Processors are Efficient

- Imagine (0.18 μm - 48 FP ALUs)
  - 3.1 W, 132 MHz, 1.5 V (meas.)
- Power dissipation is dominated (>90%) by very predictable sources
  - ALUs
  - Switches between ALUs
  - Clocks

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