Lecture 18 - Stream Processors

Mattan Erez

The University of Texas at Austin
Stream Processors Offer Efficiency and Performance

- Peak GFLOPS
- Peak DP
- mm^2/GFLOPS
- W/GFLOPS

GFLOPS

W or mm^2/GFLOP

90nm

65nm

Intel Pentium4
AMD Opteron Dual
NVIDIA G80
AMD R600
IBM Cell
Merrimac
Intel Core 2 Quad
AMD Barcelona

© Mattan Erez EE382V: Principles of Computer Architecture, Fall 2008 -- Lecture 18 (c) Mattan Erez 2008
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
  - Bulk operations and decoupling

- Software given more explicit control
  - Explicit hierarchical scheduling and latency hiding
  - Explicit parallelism
  - Explicit locality management and communication
  - Generalize streaming with bulk gather-compute-scatter

Must reduce HW “waste” but no free lunch
## Stream Processors and GPUs

### 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**
  - Needed for graphics
- **Dynamic work-loads**
  - Mostly read-only
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
• More details on HW and SW tradeoffs
  – Locality, parallelism, and scheduling
• Irregular streaming applications
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>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 16KB Cache</td>
<td>17.5</td>
<td>5.3</td>
<td>2</td>
</tr>
<tr>
<td>Transfer 64b across chip (10mm, Rep.)</td>
<td>179</td>
<td>179</td>
<td>179</td>
</tr>
<tr>
<td>Transfer 64b across chip (10mm, Cap.)</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

Leads to cache model “merging” on-chip addresses and off-chip memory
Realistic View – Only Part of Working Set is Accessible In On-Chip State
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 \textit{well in advance} of data use
  - Latency hiding
  - Blocking

- Reformulate to \textit{gather - compute - scatter}
  - Block phases into \textit{bulk operations}

- \textit{“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

• More details on HW and SW tradeoffs
  – Locality, parallelism, and scheduling

• Irregular streaming applications
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:
- $\text{FLOPs} / \text{BW}$, $\text{FLOPs} / \text{power}$, and $\text{FLOPs} / \text{area}$
Stream Processor Architecture (Merrimac)

Multiple FPUs for high-performance

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

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

Cluster switch

8 GB

DRAM bank

<64 GB/s

I/O pins

K1

K2

forces

64 64-bit MADDs (16 clusters)

cluster switch

3,840 GB/s

~61 KB

positions

64

MADDs

(16 clusters)
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

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

Somewhat specialized processors; very efficient over a range of high-performance applications
Stream Processors are Efficient

- Imagine (0.18 $\mu$m – 48 FP ALUs)
  - 3.1 W, 132 MHz, 1.5 V (meas.)
- Power dissipation is dominated (>90%) by very predictable sources
  - RFs
  - ALUs
  - switches between ALUs
  - clocks
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
- More details on HW and SW tradeoffs
  - Locality, parallelism, and scheduling
- Irregular streaming applications
SRF Decouples Execution from Memory

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