Data Parallel Architectures

EE 392C
April 3, 2003
Nuwan Jayasena & Suzanne Rivoire

Papers

- Kozyrakis and Patterson, "Vector vs. superscalar and VLIW architectures for embedded media benchmarks"
- Khailany, Dally, et al, "Imagine: Media Processing with Streams"

Multimedia Applications

- Video, audio, graphics, signal processing…
- Characteristics
  - High data-level parallelism (DLP)
  - Little global reuse
  - Lots of computation

[*Imagine*]

A smart multimedia architecture…

- Explicitly supports DLP (rather than trying to infer it)
- Memory hierarchy: uses software-managed structures rather than a traditional cache

VIRAM: Vectors for multimedia

- Exploit DLP with vector lanes (pipes)
- Have a vector register file + memory on chip and scrap the cache

VIRAM contributions

- Lots of vector registers
- Support for multiple data widths
- Vector register permutation instructions
- Virtual addressing

"Imagine: Media Processing with Streams"

- Khailany, Dally et. al.

- Key contributions
  - Stream programming model
  - Stream processor (Imagine) architecture

Stream Programming Model

- Streams are sequences of homogeneous elements (records)
- Kernels apply the same computation to all elements of its input stream(s) and produce output stream(s)
  - Exposes data parallelism among stream elements
  - Potential for instruction-level parallelism in operations applied to one stream element
- A *stream program* orchestrates the passing of data streams among a set of kernels to implement a desired application

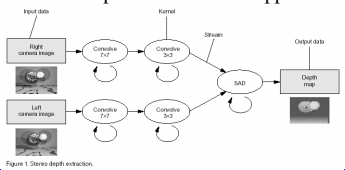


Figure 1. Stereo depth extraction.

Stream Processor (Imagine) Architecture

- Coprocessor
- Many ALUs exploit concurrency in stream applications
  - Multiple clusters in parallel exploit data parallelism (SIMD control)
  - Multiple ALUs within each cluster exploit ILP (VLIW control)
- But putting many ALUs on a chip isn't that hard
  - Feeding data to the ALUs is a much harder problem
- Bandwidth hierarchy tailored to capture locality
  - Local register files capture intermediate results between individual operations on one stream element
  - Stream register file captures intermediate streams between kernels

Streaming/Imagine Summary

- Pros
  - Programming model exposes data transfers in bandwidth hierarchy
  - Extends storage hierarchy with 2 levels of software-controlled storage
  - Record-order transfers makes DRAM accesses more efficient
- Cons
  - Programming model places more burden on programmer
  - Imagine is not a "general-purpose" processor
  - Paper doesn't provide adequate comparison points to other architectures

Advantages of Data Parallel Computing

- "Cheap parallelism"
  - Easy to express and exploit
- Predictable memory accesses
  - Decouple memory accesses from computation to cover memory latency
- Lower power for a given level of performance
  - Lower clock freq., lower supply voltages, and/or shallower pipelines, lower instruction overhead etc.
- Lower design effort
  - Replicate a "simple" pipeline or cluster
  - Lower circuit implementation effort

Key Issues for EE392C

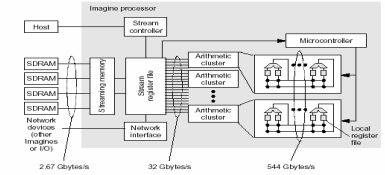
- Data parallel, on-chip multi-processors
  - Greater exploitation of DLP by running multiple processors in SIMD?
  - Exploit TLP (in addition to DLP and ILP) by each processor running its own thread?
  - Control and communication (incl. synchronization) techniques?
- Data parallel architecture as a "configuration" of polymorphic processors
  - What resources need to be configured
    - Memory hierarchy?
    - Compute units?
    - Instructions/control?
  - Which resources yield biggest gains with minimal reconfiguration overhead?

Discussion

- What are the key similarities and differences? Pros and cons?
  - In what situations might VIRAM outperform Imagine and vice versa?
- Are data-parallel architectures really the solution for multimedia applications?
  - Could superscalar or VLIW processors achieve the same goals (maybe with multimedia or other ISA extensions)?
- How do these architectures exploit ILP?
  - What about apps with little or no DLP?

Backup Slides

Communication in a Stream Processor



- LRFs can only be accessed within local cluster
- SRF bank only accessible to local cluster
- Any cluster can access any memory bank
  - Expensive communication network restricted to the level with lowest bandwidth
  - Conversely, communication requires going to the most expensive level of bandwidth hierarchy
- Inter-cluster network provides some communication among clusters

Imagine Performance Results

Table 1. Application bandwidth requirements versus Imagine's bandwidth hierarchy.

	Memory (Gbytes/s)	SRF (Gbytes/s)	Clusters (Gbytes/s)	Performance
Imagine peak	2.67	32	544	20 Gflops (40 16-bit GOPS)
Depth	0.83	21.06	263.02	12.1 GOPS (16-bit)
MPEG2	0.45	2.21	214.31	18.3 GOPS (16- and 8-bit)
QR	0.37	3.99	294.82	13.1 Gflops
Render	1.61	8.59	205.05	5.1 GOPS (16-bit and floating-point)

Table 2. Application and kernel performance.

	Arithmetic bandwidth	Power estimate (W)	Application performance
Applications			
Depth	12.1 GOPS (16-bit)	2.9	320 × 240 8-bit gray scale at 212 frames/s
MPEG2	18.3 GOPS (16- and 8-bit)	2.2	720 × 480 24-bit color at 105 frames/s
QR	13.1 Gflops	3.6	192 × 96 matrix decomposition in 1.1 ms
Render	5.1 GOPS (16-bit and floating-point)	2.9	14.9 million vertices/s (16.6 million pixels/s)
Kernels			
Discrete cosine transform	22.6 GOPS (16-bit)	2.6	34.8 ns per 8 × 8 block (16-bit)
Convolve7×7	25.6 GOPS (16-bit)	3.0	1.5 µs per row of 320 16-bit pixels
Fast Fourier transform	6.9 Gflops	3.8	7.4 µs per 1,024-point floating-point complex FFT