Mechanisms for Streaming Architectures

Stephen W. Keckler
Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin

August 22, 2003
Diversity in Streaming Programs

- High variability in attributes of:
  - Computation/memory BW ratio
  - Control structure
  - Memory access: data stream, constants

- Examples
  - DSP/multimedia
    - Subwork parallelism, predictable control flow
  - Scientific
    - Vectorizable with regular or irregular data access
  - Communication (packet processing, encryption, etc.)
    - Regular control flow
    - Irregular control, data dependent control flow
  - Graphics
    - Regular control/data (rasterization)
    - Irregular control/data (shading)
Challenges for Streaming Architectures

- Supporting regular and irregular control structures
  
  \[
  \text{for}(i=0;\ i<n;\ i++) \{
  \text{read}(r[i],g[i],b[i]);
  \text{Y} = K1*r+K2*g+K3*b;
  \text{I} = K4*r+K5*g+K6*b;
  \text{Q} = K7*r+K8*g+K9*b;
  \text{store}(\text{Y}[i],\text{I}[i],\text{Q}[i]);
  \}
  \]

- Different types of storage demands
  - Stream/vector register files (regular access)
  - Scalar constants
  - Indexed tables

- Scaling of kernel processors
  - Large bypass networks, instruction broadcasting
  - Amenable to pipelining – but decreases agility

- Traditional streaming architectures ⇒ regular
## DLP Control Characteristics

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Kernel Size (# instr)</th>
<th>Record Size (bytes)</th>
<th>ILP</th>
<th>Comp/Mem</th>
<th>Iterations in inner loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert</td>
<td>15</td>
<td>24</td>
<td>5</td>
<td>2.5</td>
<td>0</td>
</tr>
<tr>
<td>DCT</td>
<td>1728</td>
<td>512</td>
<td>6</td>
<td>27</td>
<td>16</td>
</tr>
<tr>
<td>High pass filter</td>
<td>17</td>
<td>72</td>
<td>3.4</td>
<td>1.7</td>
<td>0</td>
</tr>
<tr>
<td>FFT</td>
<td>10</td>
<td>48</td>
<td>3.3</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>LU</td>
<td>2</td>
<td>16</td>
<td>1</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>MD5</td>
<td>680</td>
<td>80</td>
<td>1.6</td>
<td>56.7</td>
<td>0</td>
</tr>
<tr>
<td>Blowfish</td>
<td>364</td>
<td>8</td>
<td>2.0</td>
<td>182</td>
<td>16</td>
</tr>
<tr>
<td>Rijndael</td>
<td>650</td>
<td>16</td>
<td>11.8</td>
<td>162.5</td>
<td>10</td>
</tr>
<tr>
<td>Vertex_simple_light</td>
<td>95</td>
<td>56</td>
<td>4.3</td>
<td>7.3</td>
<td>0</td>
</tr>
<tr>
<td>Fragment_simple_light</td>
<td>64</td>
<td>64</td>
<td>3.0</td>
<td>5.3</td>
<td>0</td>
</tr>
<tr>
<td>Vertex_reflection</td>
<td>94</td>
<td>72</td>
<td>7.1</td>
<td>8.5</td>
<td>0</td>
</tr>
<tr>
<td>Fragment_reflection</td>
<td>98</td>
<td>40</td>
<td>6.2</td>
<td>12.3</td>
<td>0</td>
</tr>
<tr>
<td>Vertex_skinning</td>
<td>112</td>
<td>128</td>
<td>6.8</td>
<td>4.5</td>
<td>Variable</td>
</tr>
</tbody>
</table>

### Benchmarks
- **Multimedia:** Convert, DCT, High pass filter, FFT, LU, MD5, Blowfish, Rijndael
- **Scientific Network:** Vertex_simple_light, Fragment_simple_light, Vertex_reflection, Fragment_reflection, Vertex_skinning
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Read Record size (bytes)</th>
<th>Write Record size (bytes)</th>
<th># scalar constants</th>
<th>Lookup table size</th>
<th># of Table Accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert</td>
<td>24</td>
<td>24</td>
<td>9</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>DCT</td>
<td>512</td>
<td>512</td>
<td>10</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>High pass filter</td>
<td>72</td>
<td>8</td>
<td>9</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>FFT</td>
<td>48</td>
<td>32</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>LU</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>MD5</td>
<td>80</td>
<td>16</td>
<td>65</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Blowfish</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>256</td>
<td>82</td>
</tr>
<tr>
<td>Rijndael</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>Vertex_simple_light</td>
<td>56</td>
<td>48</td>
<td>32</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Fragment_simple_light</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Vertex_reflection</td>
<td>72</td>
<td>16</td>
<td>35</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Fragment_reflection</td>
<td>40</td>
<td>24</td>
<td>7</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Vertex_skinning</td>
<td>128</td>
<td>72</td>
<td>32</td>
<td>288</td>
<td>36</td>
</tr>
</tbody>
</table>
Desirable Attributes of a Stream Processor

- Performs well on DLP programs with different attributes
  - Synchronous core to minimize synchronization overheads on traditional vector/stream applications
  - MIMD-like capabilities for applications with irregular control
  - Support for different types of data structures

- Partitioned and scalable microarchitecture
  - Dataflow instruction execution
  - Limit/eliminate global broadcast of instructions/data

- Decoupled processor core
  - From memory system to enable memory fetch parallelism
  - From other processor cores to enable kernel pipelining

- Efficient instruction distribution and re-use
  - Exploit spatial/temporal locality
Dataflow Execution in TRIPS Core

- **SPDI: Static Placement, Dynamic Issue**
  - Instructions execute in dataflow order
- Instructions stream in from left, data from right
- ALU Chaining
- Pipeline instruction distribution and execution
- Static unrolling for latency tolerance
Instruction Fetch Efficiency

• Spatial loop unrolling
  – Copy same instruction sequence across multiple execution units

• Mapping reuse
  – Reset previously mapped instructions
  – Re-execution without refetch
Memory Accesses

- Irregular memory access
  - Map to hardware cache hierarchy
- Regular data accesses
  - Subset of L2 cache banks configured as Stream Register File (SRF)
  - High bandwidth data channels to SRF, reduced address BW
  - DMA engines transfer between SRF and DRAM or other SRFs
- Constants saved in reservation stations with corresponding instructions
MIMD Extensions

- Extensions to ALU nodes
  - Local control (PC)
    - Independent loops
    - Conditionals
  - Treat frame space as local instruction and data memory
- Tightly coupled MIMD
  - Central sequencer
    - Distributes kernel code to ALUs
    - Determines when to proceed to next kernel
  - Inter-ALU communication under investigation
- Issue: limited reservation station storage
  - Add more local instruction storage
  - Aggregate multiple ALUs into larger logical ALU
Performance Comparisons

Relative Performance

- Single-thread
- MIMD

Bars for:
- dct
- rijndael
- vertex skinning
TRIPS Chip

- 4 cores (with streaming support)
- L2 cache and SRF memory banks
- Pipelining across kernels mapped to different cores
  - Extend to system through off-chip channels
Summary

- Streaming applications: irregular and regular
  - Irregular control and data access
  - Irregularity increasing (particularly in graphics domain)
- Limitations of many stream/vector kernel processors
  - Execution model demands regular control flow
    - Clever extensions such as conditional streams
  - Poor support for irregular table access
    - Scatter/gather for sparse/irregular vectors
- Hybridization can extend range of applications
  - Efficient caching support
  - Flexible instruction execution
    - Decouple execution engines and memory
    - Single instruction stream for regular control flow, loop-carried dependencies
    - Tightly coupled MIMD for irregular control flow
    - Less synchronization means better scalability