Source from:

An Energy-Efficient Mobile Vertex Processor With Multithread Expanded VLIW Architecture and Vertex Caches

Outline

• Introduction
• Related Works
• Datapath Architecture
• Vertex Cache
• Experimental Results
• Conclusion
Introduction

• High performance vertex processor (VP)
  – Floating-point vertex processor
  – 120 Mvertices/s and 157mW @ 100 MHz

• Multithread expanded VLIW (MT-eVLIW) architecture
  – Balance and accelerate the performance

• Vertex cache mechanism with pre- and post-vertex caches
  – Reduce the memory bandwidth
Related Works (1/4)

- **Classified function Units**
  - EX: SFU, AU, MAU

- **Special function unit (SFU)**
  - Exponent, reciprocal, logarithm, square root

- **Adder unit (AU)**
  - Addition, subtraction, etc.

- **Multiply-array unit (MAU)**
  - Multiplication, dot-production, etc.

- MAU occupies 77.8% of total instructions.
• Single-thread SIMD (ST-SIMD) datapath

Fig. 2. Conventional ST-SIMD datapath architecture of the vertex shader.
Related Works (3/4)

- Single-threaded (ST)
  - Long latency and data hazard
- Multi-threaded (MT)
  - Inter-vertex independency
  - Area overhead
    - Input/output register set for each thread

![Diagram](image)

(a) Single thread

(b) Four threads

(period of 4-threads  
cycle improvement)
• Multi-threaded VLIW (MT-VLIW)
  – Intra-vertex independency
    • A lot of independent inst. groups in vertex program
    • Use idling functional units in parallel
  – Still have idle time because of instruction inclination
    • Ex: { DP4 dst0.x src0 src1;
      DP4 dst0.y src0 src2;
      DP4 dst0.z src0 src3;
      DP4 dst0.w src0 src4; }

• Multi-threaded VLIW with expanded functional units (MT-eVLIW)
  – Only expand frequently used function unit → MAU
Datapath Architecture

• Input and output registers
• Operand provider and writeback organizer
• Vertex multi-threading
• Four-issue MAU expanded VLIW datapath
  – 4 multiply-array unit (MAU)
  – 1 adder unit (AU)
  – 1 special function unit (SFU)
Input and Output Registers

• Vertex Attributes Buffer (VAB)
  – A temporal buffer between the processor core and the input vertex buffer, a pre-vertex cache.
  – Store vertex position, color, normal, texture coordinate, and etc.
  – 1r port

• Wide CMEM (wCMEM)
  – Store constants for geometry transformation
  – Four consecutive operands by a single address
  – 1r => 4r

• General Purpose Register (GPR)
  – 3r/1w => two half 2r/1w
Wide CMEM. A 256-entry single CMEM is divided into four banks to provide consecutive operands by a single address.
The 3r/1w GPR configurations are changed to a pair of 2r/1w half-GPR configuration. (a) Four 3r/1w GPRs for each thread (4 × 32 entries); (b) one large 3r/1w GPR (1 × 128 entries); (c) two half 2r/1w GPRs (2 × 64 entries).
Operand Provider and Writeback Organizer

• The large number of operands, results, and their instruction cause significant overhead in terms of power and area of the on-chip memories.

• Minimize the number of registers
  – Sharing the operands
  – Re-allocating the writeback paths

• The operand provider fetches 7 different operands with four source addresses.
  – 4 operands from one address for wMEM
  – 3 operands from three addresses for VAB and GPR
Multi-issue Example Operation

- **Input vertices**
  - IMEM[PC]
    - Instruction Fetching
      - Issue 0: MUL r1.x r0 c2
      - Issue 1: MUL r1.y v3 r0
      - Issue 2: ADD r1.z r0 r3
      - Issue 3: LOG vo2.w r3
    - 4 read operands
    - Operand Fetching
      - GPR[r0], GPR[r3], CMEM[c2], VAB[v3]
- **Operand Distribution**
  - [r0][c2] [r0][v3] [r0][r3] [r3]
    - Input Operands
    - [r1.x] [r1.y] [r1.z] [vo2.w]
    - 4 results
    - Destination Re-allocation
      - Writeport 0 to GPR
      - Writeport 1 to VOB

- **Output Operands**
  - MAU0
  - MAU1
  - AU
  - SFU
  - r1
  - vo2
Vertex Multi-Threading (1/2)

- Decrease latency
- Four thread processing
Vertex Multi-Threading (2/2)

![Diagram of time slot of a thread showing execution stages: IF, ID, OF, EX0, EX1, EX2, EX3, WB. The next value of PC can be decided at the slots s0 to s3. Branch target address is decided at the slot s2 without any penalties. Inst. [n] of vertex[n] and Inst. [n+1] of vertex[n] are also shown.]
Datapath Architecture (2/2)

• 4 MAU with 32x4x4-bit floating-point operation
  – 77.8% instruction allocation among test benches.
  – Each MAU has quad floating-point SIMD which processes four 32-bit floating-point scalars: \{r, g, b, a\} and \{x, y, z, w\}.

• 1 AU with 32x4-bit floating-point operation

• 1 SFU with 32x1-bit floating-point operation

• **Single cycle transformation can be achieved.**

• Four instructions via VLIW in a single cycle with 128-bit width
**Vertex Cache**

- **Pre-TnL cache (Before computing TnL)**
  - In triangle strip, the vertices would be reused.
  - Reduce bandwidth
  - Improve performance
  - Improve power consumption

- **Post-TnL cache (After computing TnL)**
  - A vertex will remain the same after TnL in a frame.
  - Also called VOB
  - Reduce bandwidth
  - Improve performance
  - Improve power consumption

* TnL: Transformation and Lighting
Pre-TnL Vertex Cache

• After vertex fetch, the vertex about to be processed (TnL) is placed in Pre-TnL cache.
• The Pre-TnL cache which has 32 entries can get 81% hit rate.
Post-TnL Vertex Cache

- After TnL, the result is saved in post TnL cache.
- In this analysis, the post TnL cache which has 16 entries gets the best performance gain.
Cache Placement

Result 1

Host

Internal Bus

Host IF

Pre cache (32-entry)

DM

Vertex Shader

Post cache (8-entry)

next index

vertex data

post cache miss

data request

Hit

processed vertex data

Hit

processed vertex data

External memory controller

NCTU VLSI Information Processing Research Lab
Bandwidth Reduction (1/2)

[pre vertex cache hit/miss, post vertex cache hit/miss].

- **[hit, hit]:**
  - No external memory access (i.e., bandwidth saving), no vertex fetch to input buffer (i.e., performance improvement).

- **[hit, miss]:**
  - No external memory access, vertex fetch to input buffer and TnL processing of that vertex.

- **[miss, hit]:**
  - No external memory access, no vertex fetch to input buffer.

- **[miss, miss]:**
  - External memory access, vertex fetch to input buffer after filling the corresponding pre TnL cache slot, and then TnL processing of that vertex.

Up to 65% of data bandwidth between the host and the processor can be reduced.
Bandwidth Reduction (2/2)

• The improvement of pre-cache (32-entry)
  – Reduce 32.8% bandwidth by reusing fetched vertices
• The improvement of post-cache (8-entry)
  – Enhance 16.7% performance
• All together
  – Reduce the 65% data bandwidth between host and the processor
TABLE I
CHARACTERISTICS OF THE VERTEX PROCESSOR

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process technology</td>
<td>0.18-μm 1P4M CMOS</td>
</tr>
<tr>
<td>Chip size</td>
<td>4 mm × 4 mm (core)</td>
</tr>
<tr>
<td></td>
<td>5 mm × 5 mm (chip)</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>1.875 V</td>
</tr>
<tr>
<td>Frequency</td>
<td>100 MHz</td>
</tr>
<tr>
<td>Power dissipation</td>
<td>231.8 mW (measured @ 1.875 V)</td>
</tr>
<tr>
<td></td>
<td>157 mW (simulated @ 1.8 V)</td>
</tr>
<tr>
<td>Transistor counts</td>
<td>1.5 M logic, 22 kB on-chip SRAM</td>
</tr>
<tr>
<td>Performance</td>
<td>2.5 GFLOPS *1)</td>
</tr>
<tr>
<td></td>
<td>120 Mvertices/s *2)</td>
</tr>
<tr>
<td>Supported shader standards</td>
<td>Vertex Shader Model 3.0 *3)</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 2.0 *4)</td>
</tr>
<tr>
<td>Enhanced hardware features</td>
<td>Four-issue VLIW architecture</td>
</tr>
<tr>
<td></td>
<td>Four-threaded floating-point datapath</td>
</tr>
<tr>
<td></td>
<td>Single-cycle geometry transformation</td>
</tr>
<tr>
<td></td>
<td>Hazard-free datapath by multi-threading &amp; forwarding</td>
</tr>
<tr>
<td></td>
<td>Pre &amp; post TnL vertex caches</td>
</tr>
<tr>
<td></td>
<td>Penalty-free static/dynamic branching</td>
</tr>
</tbody>
</table>

*1) peak floating-point performance
*2) measured in the case of a geometry transformation
*3) supports the VS3.0 except for the texture lookup instruction, “texldl”
*4) supports the vertex shader standard of OpenGL ES 2.0 except for the texture lookup
• Demo platform
Comparison Result (1/4)

- Performance comparison on various geometry operations
Comparison Result (2/4)

- Average energy dissipation
Comparison Result (3/4)

### TABLE II
PORTIONS OF THE SHARED OPERANDS AND THE RE-ALLOCATED WRITEBACK PATHS

<table>
<thead>
<tr>
<th># of bench</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>4.2</td>
<td>3.0</td>
<td>5.1</td>
<td>2.9</td>
<td>4.1</td>
<td>2.4</td>
<td>4.3</td>
<td>2.3</td>
<td>3.5</td>
<td>3.1</td>
<td>2.7</td>
<td>8.0</td>
<td>3.8</td>
</tr>
<tr>
<td>Ns (%)</td>
<td>26.0</td>
<td>15.2</td>
<td>30.4</td>
<td>12.5</td>
<td>24.1</td>
<td>11.3</td>
<td>23.1</td>
<td>12.5</td>
<td>20.0</td>
<td>23.5</td>
<td>14.0</td>
<td>37.5</td>
<td>20.8</td>
</tr>
<tr>
<td>Nw</td>
<td>2.5</td>
<td>1.7</td>
<td>2.8</td>
<td>1.7</td>
<td>2.6</td>
<td>1.5</td>
<td>3.0</td>
<td>1.6</td>
<td>2.1</td>
<td>2.0</td>
<td>1.8</td>
<td>4.0</td>
<td>2.3</td>
</tr>
<tr>
<td>Nr (%)</td>
<td>46.7</td>
<td>25.0</td>
<td>56.0</td>
<td>23.8</td>
<td>50.0</td>
<td>27.5</td>
<td>66.7</td>
<td>18.2</td>
<td>42.9</td>
<td>27.3</td>
<td>27.6</td>
<td>75.0</td>
<td>40.5</td>
</tr>
</tbody>
</table>

No : Average number of used operands per single instruction slot.

Ns : The rate of the shared operands among the total required operands.

Nw : The average number of instructions per single instruction slot.

Nr : average rate of the writeback re-allocations per instruction slot.
Comparison Result (4/4)

### Vertex Processing Performance

<table>
<thead>
<tr>
<th></th>
<th>Mvertices/s</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sohn [3]</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arakawa [4]</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kim [5]</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Comparison Table

<table>
<thead>
<tr>
<th></th>
<th>Func. Unit</th>
<th>Voltage</th>
<th>Process</th>
<th>Freq.</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim [5]</td>
<td>GE+RE</td>
<td>1.2 V</td>
<td>0.13μm</td>
<td>166 Mhz</td>
<td>407 mW</td>
</tr>
<tr>
<td>Arakawa [4]</td>
<td>GE only</td>
<td>1.25 V</td>
<td>0.13μm</td>
<td>400 Mhz</td>
<td>250 mW</td>
</tr>
<tr>
<td>Sohn [3]</td>
<td>GE+RE</td>
<td>1.8 V</td>
<td>0.18μm</td>
<td>200 Mhz</td>
<td>155 mW</td>
</tr>
<tr>
<td>This work</td>
<td>GE only</td>
<td>1.875 V</td>
<td>0.18μm</td>
<td>100 Mhz</td>
<td>231.8 mW</td>
</tr>
</tbody>
</table>
Conclusion

- Propose an energy efficient datapath architecture
  - Multi-threaded VLIW with expanded functional units (MT-eVLIW)
- Vertex caches for the vertex processor making use of the specific characteristics
  - Pre TnL cache reduces the data bandwidth between the host and the vertex processor
  - Post TnL cache improves the performance and the energy consumption due to reusing TnL results.
References

