The Trimedia TM-1 PCI VLIW Media Processor

Gerrit A. Slavenburg
Selliah Rathnam
Henk Dijkstra

1996 (eighth) Hot Chips Symposium
final slides - July 19, 1996
TM-1 Block Diagram

System on a chip

for system details, see Proceedings of the 1995 Micro Processor Forum.

Applications
- standalone / PC
- video conferencing
- settop decoder
- DVD decoder
- 3-D graphics
- audio en/de/transcode (AC-3, MPEG)
- audio synthesis

Applications

Philips Semiconductors -- TriMedia
TM-1 VLIW engine

- 5 RISC operations/cycle
- 32 kByte Icache (compressed)
- dual port 16 kByte Dcache
- conditional (guarded) operations
  \[ R_g : R_{dest} = \text{imul} R_{src1}, R_{src2} \]
- multimedia operation set

<table>
<thead>
<tr>
<th>Func Type</th>
<th>Qty</th>
<th>Latency</th>
<th>Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ALU</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Memory</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Shift</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DspAlu</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DspMul</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Branch</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Falu</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Ifmul</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Fcomp</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Fdiv/Fsqrt</td>
<td>1</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>
TM-1 cache architecture

- I-cache contains compressed VLIW instructions
- 32 kByte I-cache, 8 way set associative, block size 64 bytes
- 16 kByte D-cache, 8 way set associative, block size 64 bytes
- Advanced D-cache features:
  - Copyback, allocate on write, non-blocking cache
  - 11 cycle read-miss, 3 cycle write-miss penalty
  - (pseudo) dual ported
  - Streaming
  - Dual entry copyback buffer
  - Programmer controlled prefetching
  - Programmer controlled alloc
  - Optional cache locking
  - Performance evaluation support
TM-1 MPEG-2 decoder stats

DVD-batman bitstream, variable rate, 4 - 9 Mbit/sec
NO programmer prefetch/alloc
100 MHz TM-1 CPU load for video decoding = 64 %
averged 3.95 useful RISC operations/VLIW operation

<table>
<thead>
<tr>
<th>CPI contributor</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction issue</td>
<td>1.00</td>
</tr>
<tr>
<td>I-cache misses</td>
<td>0.07</td>
</tr>
<tr>
<td>D-cache misses</td>
<td>0.27</td>
</tr>
<tr>
<td>D-cache bank conflict stalls</td>
<td>0.03</td>
</tr>
<tr>
<td>total CPI (clock cycles/VLIW instruction)</td>
<td>1.37</td>
</tr>
</tbody>
</table>

TOTAL CPU + memory system performance 3.95/1.37 = 2.9 ops/clock

(the majority of these operations are SIMD multi-media operations used in the DCT and Motion Compensation code)
Problem: DVD video excursions. 3 runs, 44 out of 500 images exceed 85% CPU load. Frame 2-9 (91 %), 109-118 (95 %), 315-340 (88 %).

Solution: quality degradation (will disappear with prefetch & faster TM-1’s)

---

**DVD decoder resource usage overview (100 MHz)**

<table>
<thead>
<tr>
<th></th>
<th>CPU load</th>
<th>SDRAM load (= highway load)</th>
<th>effective ops/ cycle^a</th>
</tr>
</thead>
<tbody>
<tr>
<td>program stream decoder</td>
<td>&lt; 5 %</td>
<td>&lt; 2 %</td>
<td>no recent data</td>
</tr>
<tr>
<td>video decoder^b</td>
<td>64 %</td>
<td>18 %</td>
<td>2.9</td>
</tr>
<tr>
<td>audio decoder^c</td>
<td>4.6 %</td>
<td>0.95 %</td>
<td>2.4</td>
</tr>
<tr>
<td>subpicture decode/insert</td>
<td></td>
<td></td>
<td>no data available yet, code under development</td>
</tr>
<tr>
<td>VBI decode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI decode</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^a speedup over 1 issue/cycle machine divided by CPI due to cache misses. Effective ops include multimedia ops.
^b DVD-Batman stream, 4-9 Mbit/sec., first 500 images, no programmer prefetch
^c 44.1 ksample/sec stereo 16 bit MPEG L2 audio
TM-1 performance sample : 3-D setup (1)

input:
- description of the 3 vertices of a triangle in 3-D
- per vertex: screen coordinates, 4 color values, Z, 1/w, u/w and v/w
  (totalling 10 float values/vertex, or 30 float values total)

output:
- 44 fixed point values per triangle to off-chip raster engine

computation:
- sort vertices by Y coordinate, compute all derivatives
- 4 float divisions
- 83 float multiplications
- 14 float comparisons
- 93 other float operations (fadd, fsub, fixpoint convert)
- 24 integer multiply operations
- total of 410 operations, incl. loads/stores
TM-1 performance sample: 3-D setup (2)

Table 1: 3D triangle setup duration in CPU clock cycles/triangle

<table>
<thead>
<tr>
<th></th>
<th>TM-1 (cycles)</th>
<th>P5-133 (cycles)(^a)</th>
<th>P6-200 (cycles)</th>
<th>TM-1 over P5</th>
<th>TM-1 over P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (no multimedia ops)</td>
<td>210</td>
<td>2526</td>
<td>2440</td>
<td>12x</td>
<td>12x</td>
</tr>
<tr>
<td>CPU (with multimedia ops)</td>
<td>96(^b)</td>
<td>2526(^c)</td>
<td>2440</td>
<td>26x</td>
<td>25x</td>
</tr>
<tr>
<td>CPU (with multimedia ops) + memory system</td>
<td>170(^d)</td>
<td>3591</td>
<td>3480</td>
<td>21x</td>
<td>20x</td>
</tr>
</tbody>
</table>

- \(^a\) Visual C++ 4.0 ‘maxspeed’ optimized, L2 cache 256 kBytes, 32 MByte DRAM
- \(^b\) this code achieves 4.68 useful operations/cycle
- \(^c\) MMX does not speedup triangle setup computation
- \(^d\) no programmer prefetch
## Architectural statistics

(min/avg/max)

<table>
<thead>
<tr>
<th>application category</th>
<th>#1 dyn % of guarded ops&lt;sup&gt;a&lt;/sup&gt;</th>
<th>#2 fine grain par. (speedup)&lt;sup&gt;b&lt;/sup&gt;</th>
<th>#3 memory system (CPI)&lt;sup&gt;c&lt;/sup&gt;</th>
<th>#4 effective ops per cycle #2/#3&lt;sup&gt;d&lt;/sup&gt;</th>
<th>#5 SDRAM bandwidth util.%&lt;sup&gt;e&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>general purpose</td>
<td>23%/26%/30%</td>
<td>1.41/2.46/3.61</td>
<td>1.03/1.37/2.39</td>
<td>1.13/1.87/2.59</td>
<td>3%/21%/66%</td>
</tr>
<tr>
<td>MPEG video decode</td>
<td>14.3%</td>
<td>3.95</td>
<td>1.37</td>
<td>2.88</td>
<td>18%</td>
</tr>
<tr>
<td>MPEG audio decode</td>
<td>5.2%</td>
<td>4.08</td>
<td>1.70</td>
<td>2.40</td>
<td>0.95%</td>
</tr>
<tr>
<td>3-D workload</td>
<td>17%/18%/19%</td>
<td>3.67/4.10/4.68</td>
<td>1.05/1.41/1.77</td>
<td>2.64/2.98/3.49</td>
<td>2%/24%/46%</td>
</tr>
</tbody>
</table>

<sup>a</sup> operations with dynamically computed guard as a percentage of total operations executed - shows frequent use of conditional execution

<sup>b</sup> execution cycles on RISC (1 op/cycle) with same instruction set/execution cycles on TM-1 VLIW CPU - without cache effects

<sup>c</sup> Cycles Per Instruction = total # clock cycles / total number of VLIW instructions executed - shows quality of cache system

<sup>d</sup> dividing column 2 by column 3 results in actual RISC instruction equivalents per clock cycle (CPU + memory system)

<sup>e</sup> memory utilization. 100% corresponds to 400 MByte/sec SDRAM bandwidth utilization
TM-1 silicon status

CPU Test Chip:
- DSPCPU, caches, SDRAM i/f, timers, Vin, PCI i/f, 80 MHz issue rate
- CTC 1.0 silicon (0.5 u 4LM CMOS) selective sampling May 96
- CTC 1.1 silicon (0.5 u 4LM CMOS) sampled to EAP customers Jul 96
- 3.3 Volt, 2.5 Watt (operating)

TM-1 chip:
- complete TM-1 system
- 0.50 u 4LM CMOS 100 MHz issue rate samples Oct 96
- 0.50 u 4LM CMOS 100 MHz RFS (volume availability) Q1 97
- 0.35 u 4LM CMOS 132 Mhz issue rate samples Q1 97
- 3.3 Volt, 4 Watt

TM-1c
- 0.35u 4LM 150 MHz compacted, pin-compatible, enhanced version
  samples Q3 97
TM-1 programming

- Open architecture, 3’rd party application development encouraged
- pSOS+M real-time kernel
- State-of-the-art fine-grain parallelizing C and C++ compilers available to EAP customers today with CTC boards
- Multi-media libraries with well-tuned code available with TM-1 RFS
- Complete application software available with TM-1 RFS
  - MPEG-1 decoder
  - MPEG-2 program stream decoder
  - 3-D graphics pipeline
  - PC audio synthesis (FM, wavetable)
  - V.34 modem
- Available in 1997
  - H.324 (PSTN) PC video conferencing Q1 97
  - H.320 (ISDN) PC video conferencing Q4 97
Lessons learned

- 21 known bugs in CTC1.0, *none* of which in the VLIW CPU and memory system
  - VLIW’s are datapath intensive, very low control area/complexity
  - directed tests + random generation of tests works well for CPU/memory system verification
  - no equivalent methodology known for verifying peripherals/co-processors and PCI
- the ratio between average CPU load and peak CPU load for software real-time video decoding is greater than expected (hence, software video decoders require overpowered CPU’s and then mix well with ‘best effort’ compute intensive tasks such as 3-D)
- VLIW’s work extremely well for signal processing and 3-D tasks, and quite good for general purpose programming
- programmer use of prefetch/alloc is very hard