How Computers Calculate Square Root?

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California

Email: anakano@usc.edu

Demystifying mathematical-function black box
Basic Computer Architecture

M. M. Mano, *Computer System Architecture* (Prentice-Hall)
FLOATING-POINT UNIT DESIGN

USING TAYLOR-SERIES EXPANSION ALGORITHMS

by

Taek-Jun Kwon

Thesis Proposal

UNIVERSITY OF SOUTHERN CALIFORNIA
ELECTRICAL ENGINEERING

September 2006
How Time Consuming Is SQRT()?

Table 1.1 Summary of prototype FPUs

<table>
<thead>
<tr>
<th>Design</th>
<th>Cycle time (ns)</th>
<th>Latency/Throughput (cycles/cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 21164 Alpha AXP</td>
<td>2.0</td>
<td>4/1, 4/1, 22-60/22-60, N/A</td>
</tr>
<tr>
<td>Hal Sparc64</td>
<td>6.49</td>
<td>4/1, 4/1, 8-9/7-8, N/A</td>
</tr>
<tr>
<td>HP PA 7200</td>
<td>7.14</td>
<td>2/1, 2/1, 15/15, 15/15</td>
</tr>
<tr>
<td>HP PA 8000</td>
<td>5.0</td>
<td>3/1, 3/1, 31/31, 31/31</td>
</tr>
<tr>
<td>IBM RS/6000 Power 2</td>
<td>14.0</td>
<td>2/1, 2/1, 16-19/15-18, 25/24</td>
</tr>
<tr>
<td>Intel Pentium</td>
<td>5.0</td>
<td>3/1, 3/2, 39/39, 70/70</td>
</tr>
<tr>
<td>Intel Pentium Pro</td>
<td>7.52</td>
<td>3/1, 5/2, 30/30, 53/53</td>
</tr>
<tr>
<td>MIPS R8000</td>
<td>13.3</td>
<td>4/1, 4/1, 20/17, 23/20</td>
</tr>
<tr>
<td>MIPS R10000</td>
<td>3.64</td>
<td>2/1, 2/1, 18/18, 32/32</td>
</tr>
<tr>
<td>PowerPC 604</td>
<td>5.56</td>
<td>3/1, 3/1, 31/31, N/A</td>
</tr>
<tr>
<td>PowerPC 620</td>
<td>7.5</td>
<td>3/1, 3/1, 18/18, 22/22</td>
</tr>
<tr>
<td>Sun SuperSparc</td>
<td>16.7</td>
<td>3/1, 3/1, 9/7, 12/10</td>
</tr>
<tr>
<td>Sun UltraSparc</td>
<td>4</td>
<td>3/1, 3/1, 22/22, 22/22</td>
</tr>
</tbody>
</table>

- **Latency**: How many clock cycles to compete 1 operation
- **Throughput**: Cycles before the next operation can be issued
Hardware Implementation of SQRT()

- **Newton-Raphson method**

\[
\sqrt{b} \approx Y_0 \left\{ 1 - \frac{1}{2} \left( 1 - \frac{b}{Y_0^2} \right) - \frac{1}{8} \left( 1 - \frac{b}{Y_0^2} \right)^2 - \frac{1}{16} \left( 1 - \frac{b}{Y_0^2} \right)^3 - \frac{15}{128} \left( 1 - \frac{b}{Y_0^2} \right)^4 \right\}
\]

- **Series expansion**
Simple SQRT() Routine

- **Initial Guess**

  \[ r = s^{\frac{1}{2}} \]
  \[ \approx f(s) = c_0 + c_1 s + c_2 s^2 + c_3 s^3 \]
  \[ = c_0 + s \times (c_1 + s \times (c_2 + s \times c_3)) \]

  where \( 0.1 < r^2 < 1.0 \)
  \( c_0 = 0.188030699; \) \( c_1 = 1.48359853 \)
  \( c_2 = -1.0979059; \) \( c_3 = 0.430357353 \)

- **Newton-Raphson Refinement**

  \[ \delta s \leftarrow s - f(s)^2 \]
  \[ r \leftarrow f(s) + \delta s / 2f(s) \]

SIMD/Vector Operation

- Each FMA operation can work on a set of multiple operands concurrently.
- Single-instruction multiple-data (SIMD) parallelism: An arithmetic operation is operated on multiple operand-pairs stored in vector registers, each of which can hold multiple double-precision numbers.

**Example:** Vector multiplier (VMUL) loads data from two vector registers, R1 and R2, each holding 4 double-precision numbers, concurrently performs 4 multiplications, and stores the results on vector register R3.