How Computers Calculate Square Root?

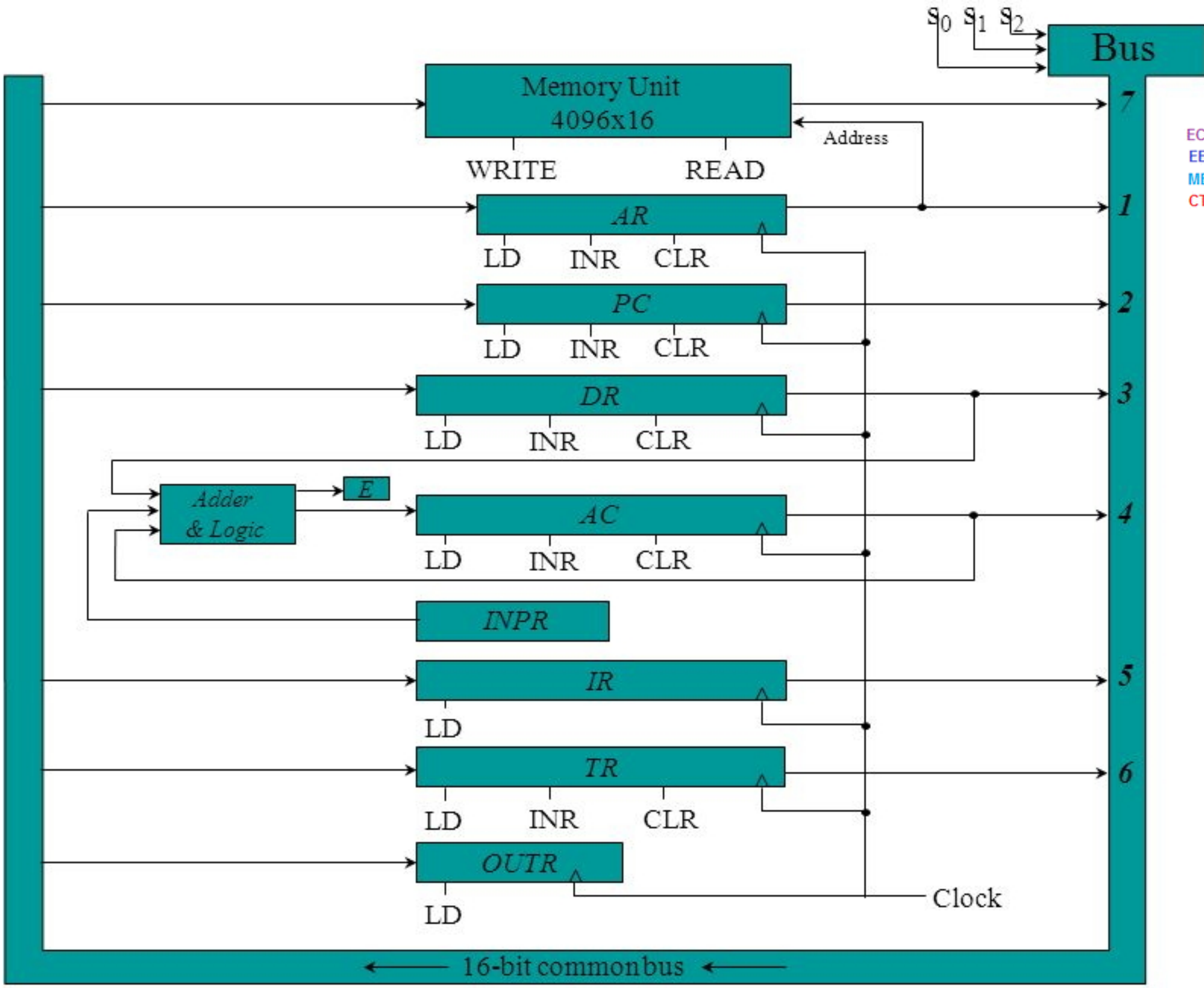
Aiichiro Nakano

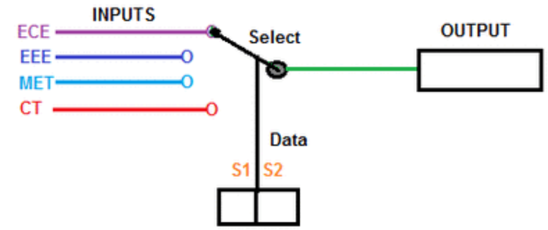
Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California

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

UNIVERISITY OF SOUTHERN CALIFORNIA
ELECTRICAL ENGINEERING

September 2006

How Time Consuming Is SQRT()?

Table 1.1 Summary of prototype FPUs

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

- **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**

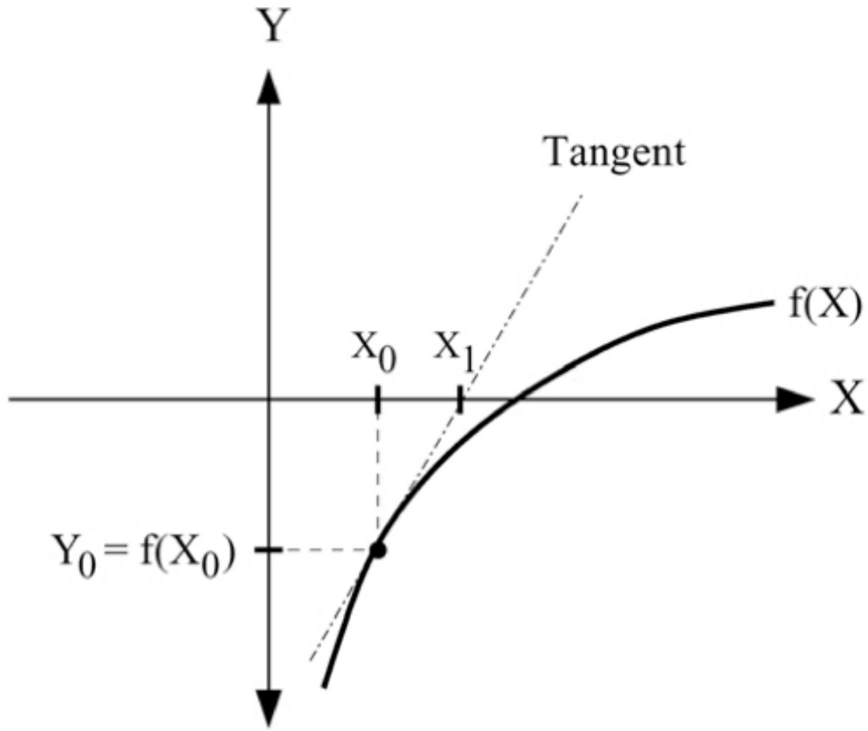


Figure 2.1 Newton-Raphson algorithm for finding the root of *f(x)*

- **Series expansion**

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

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 $0.1 < r^2 < 1.0$

$c_0 = 0.188030699$; $c_1 = 1.48359853$

$c_2 = -1.0979059$; $c_3 = 0.430357353$

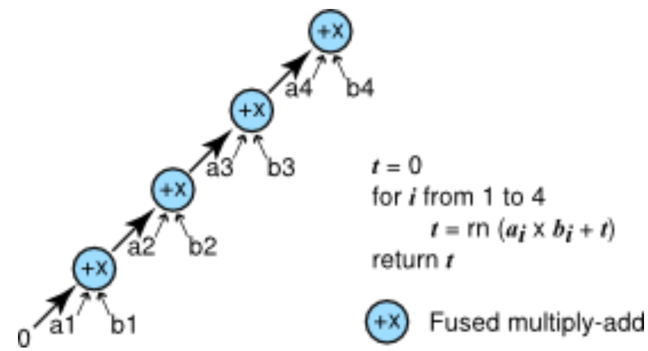
- **Newton-Raphson Refinement**

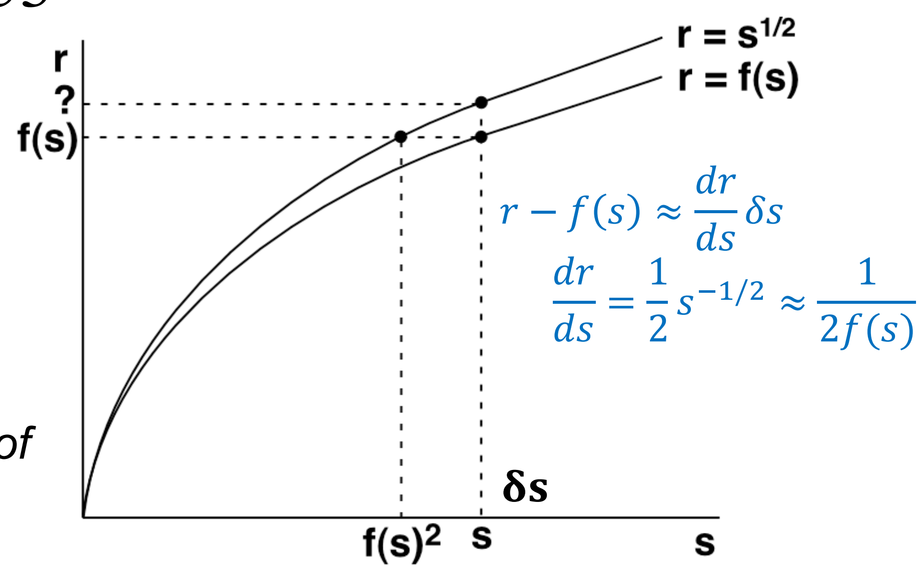
$$\delta s \leftarrow s - f(s)^2$$
$$r \leftarrow f(s) + \delta s / 2 f(s)$$

Fused multiply-add (FMA) unit

$$a \leftarrow a + b\times c$$

with 1-cycle throughput

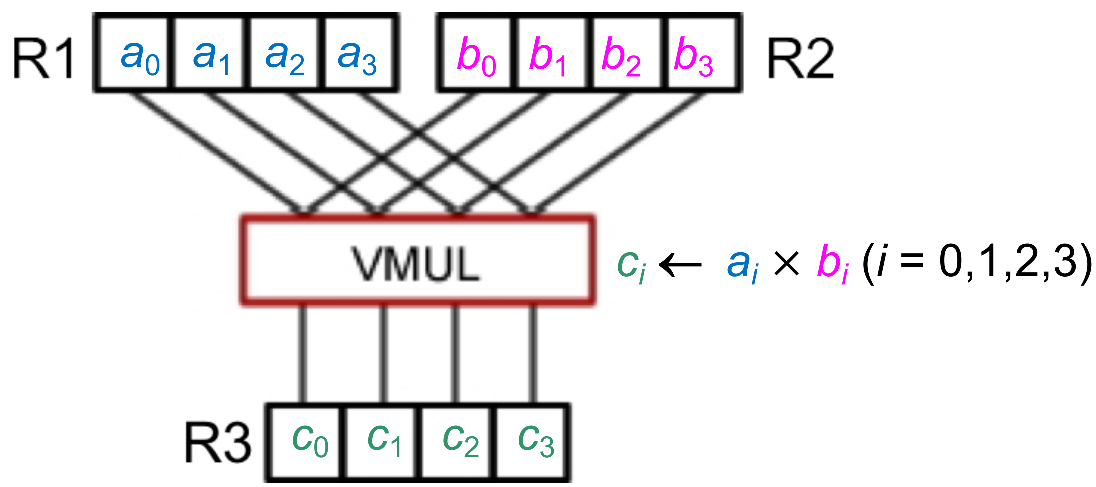




M.P. Allen & D.J. Tildesley, *Computer Simulation of Liquids* (Oxford Univ. Press, Oxford, 1987) p.143

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.