EE 486 lecture 17: What’s new in Divide.

M. J. Flynn

Two approaches

- Bipartite tables … very useful in short precision divide, as with 3D graphics.
- Higher order series … for long precision and extended precision.
- Note that both of these approaches are also useful in implementing the various HLF (higher level functions: trig, log, sqrt, etc)

Bipartite tables

- Implement first 2 terms of Taylor series for 1/b in 2 tables.
- First term is an approximation, the second term approximates the derivative, (1/b)^2
- Then b

  - First table index is b1+b2; second table index is b1+b3 (b3 defines the derivative in the region of b1).

Bipartite Tables to find (1/b)

- Based on first two terms of a Taylor series expanded about the leading bits of b, called b_h. So
- Reciprocal = (1/b_h) - Δb(1/b_h)^2 + (Δb)^2(1/b_h)^3
- Note that all terms are positive since Δb is negative.
- Use two tables, one to find the first term and one to find the second… error is approx. by the third term.

Bipartite Tables to find (1/b)

- Similar approach is to use linear (or higher order interpolation).
- Reciprocal = (1/b_h) + b_j((1/b_h)-(1/b_h+ulp))
- Now needs one table lookup then a multiply ~add.
Interpolation tables

- Can be a more general approach using a multiply and an add.
- Needs a smaller table $2^{k+2} \times (2k + 3$ or so).

Higher order divide: #1

- Now compute new dividend, $a'$ as
  - $a' = a - a_h \times (1/b_H) \times b$ and quotient
  - $q^* = q + a_h' \times (1/b_H)$ (shifted)
  - Can use redundant, $s + c$ form to speed things up.
  - Precision (m bit lookup) m-2 bits per iteration

Higher order divide: #3

- Let $b = b_H + b_L$
- Factor $1/b_H - b_L/b_H^2 + (b_L)^2(1/b_H)^3$...
- $a/(b_H + b_L) = a/b_H (1 - b_L/b_H + b_L^2/b_H^2)$
- First 2 terms $a/b = (b_H - b_L)/b_H^2$
- Look up $b_H^2$
- Precision (m bit lookup) 2m - 3 bits per iteration... can be 2m with compensation

Higher order divide (a/b)

- As with the NR on the term exam, we can use multiple terms (say $t$ terms) of the Taylor series as an iteration. So
- Reciprocal $= (1/b_H) - \Delta b(1/b_H)^2 + (\Delta b)^2(1/b_H)^3$
- $\Delta b = |b - b_H| = b_H$, all terms positive
- So look up $(1/b_H), (1/b_H)^2, (1/b_H)^3$; compute $\Delta b$ and $(\Delta b)^2$

Higher order divide: #2

- $B = (1/b_H) - \Delta b(1/b_H)^2 + (\Delta b)^2(1/b_H)^3$...
  - Look up $m$ bits of $(1/b_H), (1/b_H)^2, (1/b_H)^3$
  - Now compute new dividend, $a'$ as
  - $a' = a - a_h \times B \times b$ and quotient
  - $q^* = q + a_h' \times B$ (shifted)
  - Precision (m bit lookup) mt - t-1 bits per iteration

M. J. Flynn
Liddicoat's General Purpose Divide and Elementary Function (HLF) Unit

- Higher order series expansion can be used for really high-performance (low latency) divide and HLF units.
- Up till now we mostly used 1st order iteration with quadratic convergence.
- Higher-order iterations converge more rapidly BUT have hardware requirements.
- The parallel computation of the square, cube, and powers of an operand reduce the latency of the higher-order iteration.

Reciprocal, Square Root, and Inverse Square Root as Series Expansion

Prescaled by \( d = (1-bX) \) with \( X_0 = 1/h, Y_0 = 1/\sqrt{b} \) and \( Z_0 = \sqrt{b} \)

- **Reciprocal**
  \[
  \frac{1}{h} = h_0 (1 + d + d^2 + d^3 + \ldots)
  \]

- **Square Root**
  \[
  \sqrt{b} = b_0 (1 - \frac{1}{2d} - \frac{1}{16d^2} - \frac{1}{128d^3} + \ldots)
  \]

- **Inverse Square Root**
  \[
  \frac{1}{\sqrt{b}} = b_0 (1 + \frac{1}{2d} + \frac{3}{8} d^2 + \frac{5}{16} d^3 + \frac{35}{128} d^4 + \ldots)
  \]

Reciprocal and the Elementary Functions Represented by Taylor Series Expansions

\[
\frac{1}{h} = 1 - x + x^2 - x^3 + \ldots
\]

\[
\frac{1}{\sqrt{b}} = 1 + \frac{1}{2} x - \frac{1}{8} x^2 + \frac{1}{16} x^3 - \frac{5}{128} x^4 + \ldots
\]

\[
1/\sqrt{h} = 1 - \frac{1}{2} x + \frac{3}{8} x^2 - \frac{11}{16} x^3 + \frac{57}{128} x^4 - \ldots
\]

\[
e^x = 1 + x + \frac{1}{2} x^2 + \frac{1}{6} x^3 + \frac{1}{24} x^4 + \ldots
\]

\[
\ln(x+1) = x - \frac{1}{2} x^2 + \frac{1}{3} x^3 - \frac{1}{4} x^4 + \ldots
\]

\[
\cos(x) = 1 - \frac{1}{2} x^2 + \frac{1}{4} x^4 - \ldots
\]

\[
\sin(x) = x - \frac{1}{6} x^3 + \frac{1}{120} x^5 - \ldots
\]

\[
\arctan(x) = x - \frac{1}{3} x^3 + \frac{1}{5} x^5 - \ldots
\]

Architecture for the General Purpose Arithmetic Unit

- The powers of \( (1-bX) \) are computed in parallel.
- Latency is approximately:
  \( \approx 1 \text{ look-up table} + 3 \) \( \text{adds, sub} \)

The Parallel Squaring Unit for \((1-bX_0)^2\)

- \( x \cdot x = x^2 \):

The Parallel Cubing Unit for \((1-bX_0)^3\)

- \( x \cdot x \cdot x = x^3 \):
Hardware Structure for the Parallel Cubing Unit

Truncated PPA for 24-bit Cube
- The required cube PPA is less than 10% of a single 24-bit direct multiply!
- \( PPA_{\text{transp}} \) height = 12 bits!
- \( PPA_{\text{transp}} \) width = 8 bits!

Square PPA Column Truncation
- The divide unit was simulated for various squaring and cubing unit truncations.
- There is a knee in the curve when the squaring unit is truncated by 29 columns.
- \( E < 0.5 \) ulp with the cube \( PPA_{\text{max}} = 60 \) and square \( PPA_{\text{max}} = 31 \).

Truncated Square PPA
- The required squaring unit PPA is less than 15% of a single 24-bit direct multiply!
- \( PPA_{\text{transp}} \) height = 9 bits!
- \( PPA_{\text{transp}} \) width = 16 bits!

Final Divide Sub-unit Precision

Divide and HLF: net
- Can be done in a LUT + (1-2) MPY+ADD.
- Yes, a 4 cycle divide is possible.
- And the hardware cost is probably no more than two multipliers and an 3 way adder and (of course) a LUT.