Ansatz:

#1 Heron Iterationsformel für die Wurzel:

$b_{n+1} = 1/2 * (b_n + x/b_n)$ mit x Radikant, b_n Schätzwert, b_{n+1} Näherung der Wurzel

#2 Andere Schreibweise für die Wurzel

$x \wedge (1/2) =$ $2 \wedge (1/2 * \log_2 x)$

$1 << [1/2 * (31 - clz(x))]$ mit clz = "count leading zeros"

Problem: clz ist nur exakt für 2er-Potenzen.

Beispiel: $31-clz(64) = 6$ ($= \log_2 64$)

Lösung:

Da die 2er-Logarithmusmethode per CLZ nur für die 2er-Potenzen exakt funktioniert, ist der Wert der Lösung als "halbwegs guter Startwert" akzeptabel. Das Heronverfahren wird mit einem solchen Startwert relativ schnell zu einer Lösung konvergieren.

Rechnung:

#0 Heronverfahren beinhaltet eine Division: das ist scheiße. Division = langsam.

- Je mehr Iterationen des Heroverfahrens, desto mehr Divisonen müssen ausgeführt werden.
- Je schlechter der Startwert, desto mehr Iterationen sind nötig.

#1 Heronformel 2mal ineinander eingesetzt.

\# $b_{n+1} = 1/2 * (b_n + x/b_n) =$ $(x\wedge 2 + b_n) / (2b_n)$ //Gleichnamig

\# $b_{n+2} = (x\wedge 2 + b_{n+1}) / (2b_{n+1})$ //Eine Iteration weiter...

\# $b_{n+2} = (x\wedge 2 + 1/2 * (b_n + x/b_n)) / (2 * 1/2 * (b_n + x/b_n))$ //b_{n+1} ersetzt

\# $b_{n+2} = - b_n\wedge 3 / (x + b_n\wedge 2) + (x + b_n\wedge 2) / (4b_n) + b_n$ //Vereinfacht

\# $b_{n+2} = (x\wedge 2 + 6b_n\wedge 2 + b_n\wedge 4) / (4b_n*x + 4b_n\wedge 3)$ //Gemeinsammer Nenner

Ergebnis: Nur noch EINE Division bei ZWEI Iterationen.

#2 Machbarkeit: b_n nun in vielen Graden vorhanden (b_n^1, b_n^2, b_n^3, b_n^4)

b_n^2 = 2 ^ (1/2 * $\log_2 x$) ^ 1 == 2 ^ (1/2*1 * $\log_2 x$) = 1 << [1/2*1 * (31 - clz(x))]

b_n^2 = 2 ^ (1/2 * $\log_2 x$) ^ 2 == 2 ^ (1/2*2 * $\log_2 x$) = 1 << [1/2*2 * (31 - clz(x))]

b_n^3 = 2 ^ (1/2 * $\log_2 x$) ^ 3 == 2 ^ (1/2*3 * $\log_2 x$) = 1 << [1/2*3 * (31 - clz(x))]

b_n^4 = 2 ^ (1/2 * $\log_2 x$) ^ 4 == 2 ^ (1/2*4 * $\log_2 x$) = 1 << [1/2*4 * (31 - clz(x))]

#3 Präzision:

- Alle geraden Potenzen sind "perfekte" gute Startwerte.
- Alle ungeaden Potenzen sind "schlechte" gute Startwerte, da die Bitschiebeweite mit einem maximalen Fehler von -0.5 behaftet ist

#4 Beschränkungen:

Größte Potenz: b_n^4: 1 << [2 * (31 - clz(x))]

- Sobald der ld(x) größer als 15 ist, wird eine 32bit Schiebeoperation überlaufen
- Sobald der ld(x) größer als 31 ist, wird auch eine 64bit Schiebeoperation überlaufen

Have Fun!