

Simple Seed Architectures for Reciprocal and Square Root Reciprocal

Milos Ercegovac, Jean-Michel Muller and [Arnaud Tisserand](#)

Asilomar, October 2005

Newton-Raphson Iteration

Goal: obtain a single zero α of function f ($f(\alpha) = 0, f'(\alpha) \neq 0$)

Iteration:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Properties:

- ▷ If x_0 close enough to α , the iteration **converges** towards α
- ▷ **Quadratic** convergence \implies number of bits of accuracy roughly **doubles** after each iteration

General-purpose processors: use N.-R. or Goldschmidt iteration for division and square root

Remark: the first iterations provide small accuracy

Solution: avoid them, use initial approximations or **seeds**

Example: for 32-bit results

- ▷ x_0 with 1 bit \implies 5 iterations (1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32)
- ▷ x_0 with 8 bits \implies 2 iterations (8 \rightarrow 16 \rightarrow 32)

Number of iterations K :

seed with relative error $\leq 2^{-n}$
 result with relative error $\leq 2^{-m}$

$$K = \left\lceil \log_2 \left(\frac{m}{n} \right) \right\rceil$$

Context

Software division and square root **need hardware seeds**

Examples of Seeds

IBM 360/91, AMD K7 and Itanium

Simple Seed Architectures

- \rightarrow Reciprocal
- \rightarrow Square root reciprocal

Implementation

VHDL code generator: **seedgen**

Experimental results

Software Division and Square Root

Division: $q = a/d$

1. Evaluate $t = 1/d$ using Newton-Raphson for $f(x) = \frac{1}{x} - d$

$$x_{i+1} = x_i (2 - dx_i^2)$$

2. Compute quotient using

$$q = t \times a$$

Square Root: $s = \sqrt{c}$

1. Evaluate $r = \frac{1}{\sqrt{c}}$ using Newton-Raphson for $f(x) = \frac{1}{x^2} - c$

$$x_{i+1} = \frac{x_i}{2} (3 - cx_i^2)$$

2. Compute square root using

$$s = r \times c$$

Fast operations: need **reciprocal** and **square root reciprocal** seeds

Limitation: a seed with a large number of bits is costly to obtain

Remark: $1/x$ and $1/\sqrt{x}$ seeds also applies in the Goldschmidt \div and $\sqrt{\quad}$

Seeds Examples

IBM 360/91

Direct lookup table for $1/x$

10-bit seeds from a ROM addressed by 7 bits of the normalized divisor

AMD K7

Optimized bipartite method for $1/x$ and $1/\sqrt{x}$

14.94 bits of accuracy for the reciprocal

15.84 bits of accuracy for the square root reciprocal

Unit with 69Kb of ROM and 3 cycles of latency

Itanium

Seeds for $1/x$ and $1/\sqrt{x}$

8.886 bits of accuracy for the reciprocal (frcpa instruction)

8.831 bits of accuracy for the square root reciprocal (frsqrrta instruction)

Reciprocal Seed (1/4)

Goal: approximate $f(x) = 1/x$ with $x \in [1, 2[$ and $x = 1.x_1x_2x_3 \dots x_n$

Degree-1 minimax polynomial:

$$1.4571 - 0.5x$$

Accuracy: 4.5 bits

Modified polynomial:

$$p(x) = \frac{3}{2} - \frac{x}{2}$$

Accuracy: 3.5 bits

$x =$	0	1	\cdot	x_1	x_2	x_3	\dots	x_n	
$x/2 =$	0	0	\cdot	1	x_1	x_2	x_3	\dots	x_n
$-x/2 =$	1	1	\cdot	0	\bar{x}_1	\bar{x}_2	\bar{x}_3	\dots	\bar{x}_n
+	3/2 =	0	1	\cdot	1	0	0	0	0
$3/2 - x/2 =$	0	0	\cdot	1	\bar{x}_1	\bar{x}_2	\bar{x}_3	\dots	\bar{x}_n

+1LSB

+1LSB

Reference: M. Ito, N. Takagi and S. Yajima, *Efficient Initial Approximation for Multiplicative Division and Square Root by a Multiplication with Operand Modification*. IEEE Transactions on Computers, 1997

Minimax Polynomial Approximation

The degree- d minimax polynomial approximation to f on $[a, b]$ is the polynomial P^* that satisfies:

$$\|f - P^*\|_\infty = \min_{P \in \mathcal{P}_d} \|f - P\|_\infty$$

where \mathcal{P}_d is the set of polynomials with real coefficients and degree at most d and

$$\|f - P\|_\infty = \max_{a \leq x \leq b} |f(x) - P(x)|$$

Minimax approximations can be computed using a well-known algorithm due to Remes (available in the Maple numapprox package, for instance)

Reciprocal Seed (2/4)

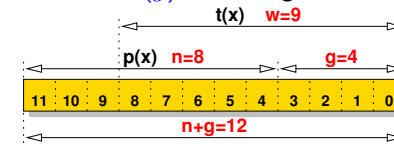
Proposed method: add a tabulated correcting term $t(x)$ to $p(x)$

$$s(x) = p(x) + t(x) = \frac{3}{2} - \frac{x}{2} + t(x)$$

Correcting term:

$$t(x) = \frac{1}{x} - \left(\frac{3}{2} - \frac{x}{2}\right)$$

Guard bits (g): rounding error



Operator architecture:

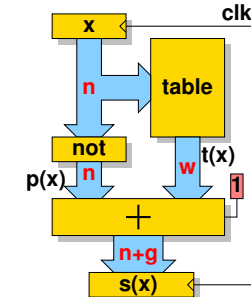
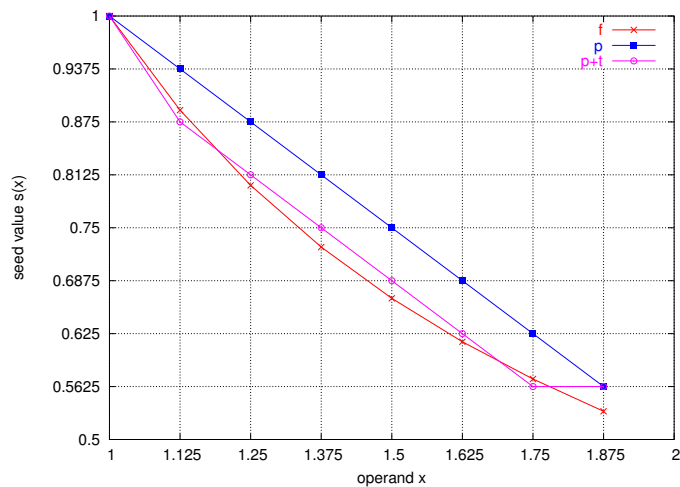


Table: $2^n \times w$ bits, adder: $n + g$ bits

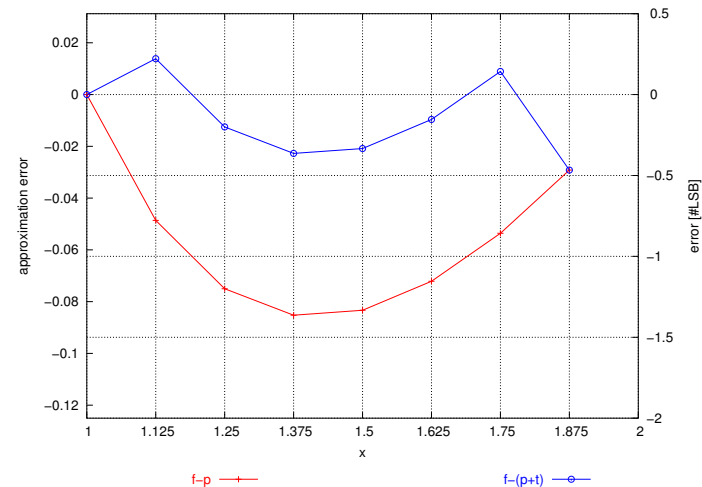
Reciprocal Seed (3/4)

Approximation results: $f = 1/x$, $n = 3$ and $g = 1$



Reciprocal Seed (4/4)

Correct rounding: $|s(x) - f(x)| < 1/2 \text{ulp}$ for $g \geq 1$



Square Root Reciprocal (1/2)

Goal: approximate $f(x) = 1/\sqrt{x}$ with $x \in [1, 2[$ and $x = 1.x_1x_2x_3 \dots x_n$

Degree-1 minimax polynomial:

$$1.2739 - 0.292x$$

Accuracy: 5.7 bits

Modified polynomial:

$$p(x) = \frac{5}{4} - \frac{x}{4}$$

Accuracy: 4 bits

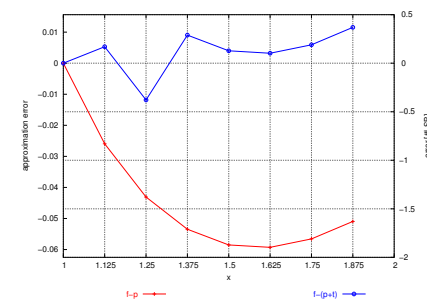
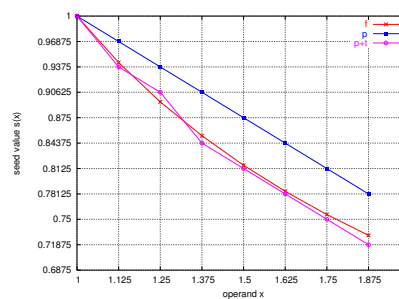
$x =$	0	1	x_1	x_2	x_3	...	x_n			
$x/4 =$	0	0	0	1	x_1	x_2	x_3	...	x_n	
$-x/4 =$	1	1	1	0	\bar{x}_1	\bar{x}_2	\bar{x}_3	...	\bar{x}_n	+1LSB
$+ 5/4 =$	0	1	0	1	0	0	0	0	0	
$5/4 - x/4 =$	0	0	1	1	\bar{x}_1	\bar{x}_2	\bar{x}_3	...	\bar{x}_n	+1LSB

Correcting term:

$$t(x) = \frac{1}{\sqrt{x}} - \left(\frac{5}{4} - \frac{x}{4}\right)$$

Square Root Reciprocal (2/2)

Approximation results and error: $f = 1/\sqrt{x}$, $n = 3$ and $g = 2$



VHDL Code Generator: seedgen

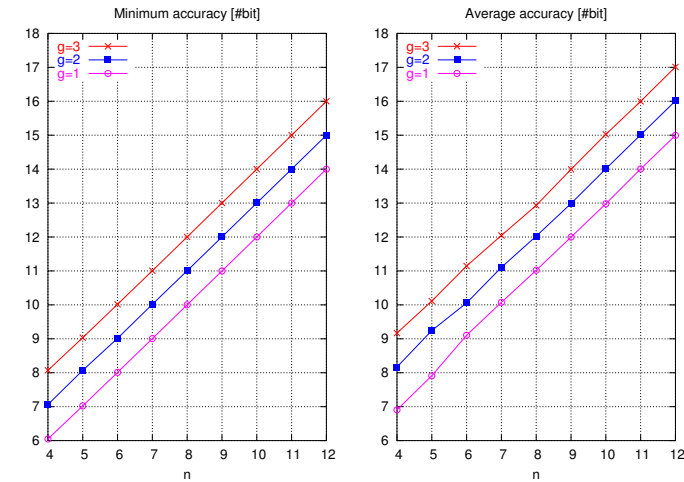
parameter	meaning	value	
		possible	default
-f	approximated function (f)	$\begin{cases} r & \text{for } \frac{1}{x} \\ s & \text{for } \frac{1}{\sqrt{x}} \end{cases}$	-
-n	argument size (n)	$2 \leq n \leq 16$	-
-g	number of guard bits (g)	$\begin{cases} 1 \leq g \leq 4 & \text{for } \frac{1}{x} \\ 2 \leq g \leq 4 & \text{for } \frac{1}{\sqrt{x}} \end{cases}$	$\begin{cases} g = 1 & \text{for } \frac{1}{x} \\ g = 2 & \text{for } \frac{1}{\sqrt{x}} \end{cases}$
-o	optimization	n for negative offset	no optimization
-r	register	$\begin{cases} o & \text{output only} \\ i & \text{input only} \\ b & \text{both input and output} \end{cases}$	no register

[seedgen](#): C program distributed under the GPL license

[web-page](http://www.lirmm.fr/~tisseran/devel/seedgen): <http://www.lirmm.fr/~tisseran/devel/seedgen>

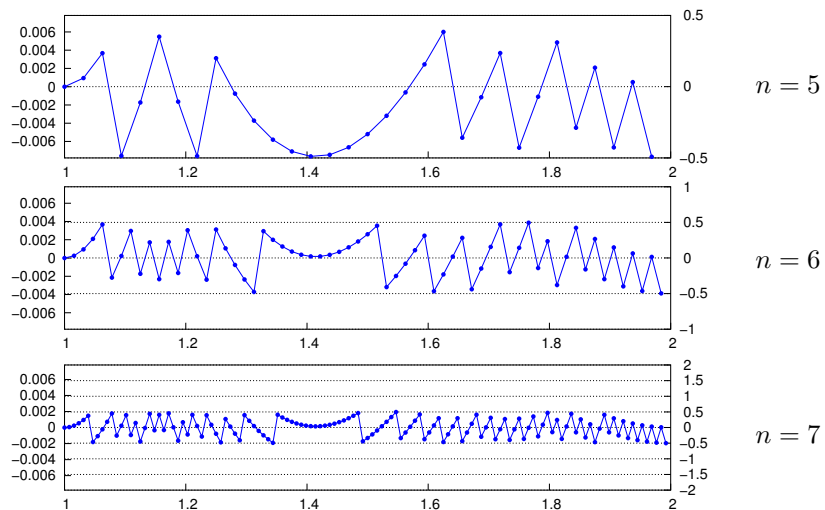
Accuracy Results (1/3)

Plots: minimum ($\geq n + g + 1$) and average ($\approx n + g + 2$) accuracy for $1/x$



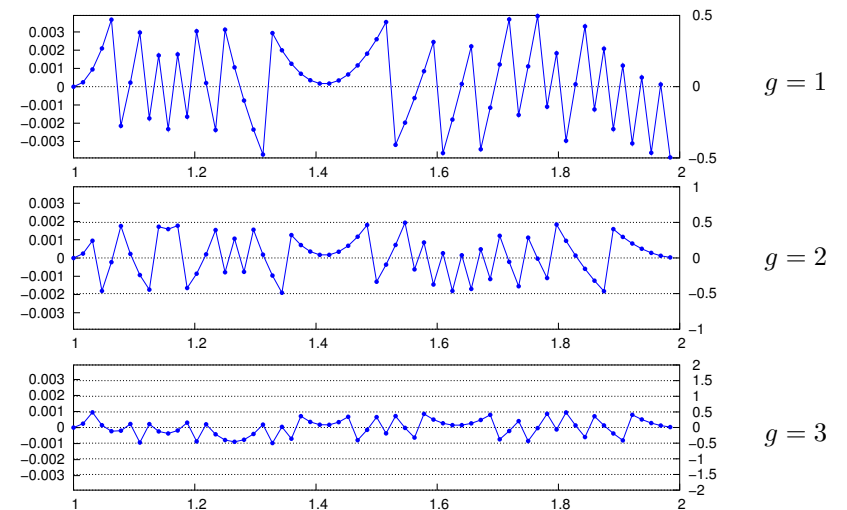
Accuracy Results (2/3)

Plot: error evolution for several values of n and the reciprocal ($g = 1$)

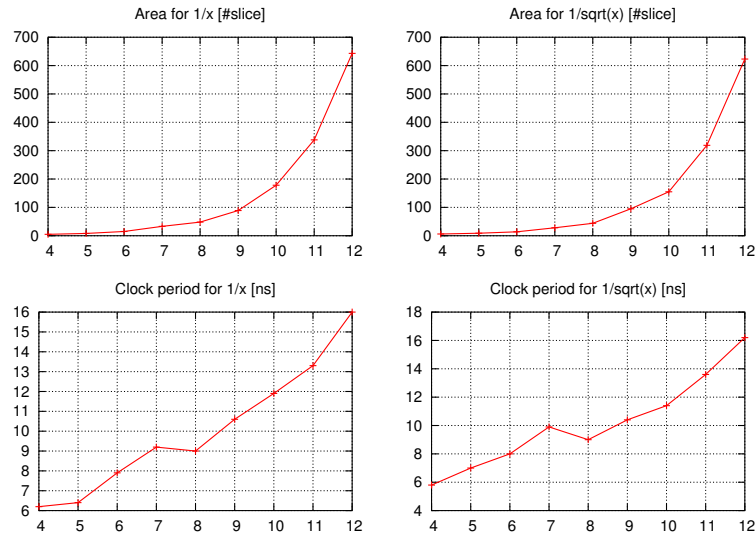


Accuracy Results (3/3)

Plot: error evolution for several values of g and the reciprocal ($n = 6$)



FPGA Implementation Results (Spartan3)



Conclusion

New seeds

reciprocal and square root reciprocal
simple architecture, very efficient for 6 to 12-bit accuracy

VHDL code generator `seedgen`

automatically generates optimized and synthesizable VHDL operators
distributed under GPL license
<http://www.lirmm.fr/~tisseran/devel/seedgen>

Future work

use new modified polynomials
use optimized multipartite tables the correcting terms

Comparison

Direct Lookup Table:

n	g	direct		optimized		seedgen	
		ROM size	area [#CLB]	area [#CLB]	period [ns]	area [#CLB]	period [ns]
7	2	1280	80	57	11.7	33	9.0
8	2	2816	176	109	12.7	48	9.2
9	2	6144	384	227	14.7	89	10.6
10	2	13312	832	448	16.4	178	11.9

Multipartite method:

up to 30% faster for the same area

References

- P. Markstein. *IA-64 and Elementary Functions : Speed and Precision*. Prentice Hall, 2000
- M. Ito, N. Takagi and S. Yajima. *Efficient Initial Approximation for Multiplicative Division and Square Root by a Multiplication with Operand Modification*. IEEE Transactions on Computers, 1997
- S. F. Oberman. *Floating Point Division and Square Root Algorithms and Implementation in the AMD-K7 Microprocessor*. 14th IEEE Symposium on Computer Arithmetic, 1999
- S. F. Anderson, J. G. Earle, R. E. Goldschmidt and D. M. Powers. *The IBM System/360 Model 91: Floating-Point Execution Unit*. IBM Journal, 1967
- F. de Dinechin and A. Tisserand. *Some Improvements on Multipartite Tables Methods*. IEEE Transactions on Computers, 2005

Questions ?

Contact

Lab:

LIRMM
CNRS, Univ. Montpellier II
161 rue Ada
34392 Montpellier cedex 5
France

Email:

arnaud.tisserand@lirmm.fr

Web-page:

<http://www.lirmm.fr/~tisseran/>

ARITH 18

2007, June 25–27
Montpellier, France

IEEE Symposium on Computer Arithmetic

- General Chair
G. Jullien
- Program Chairs
P. Kornerup
J.-M. Muller
- Local Chairs
L. Imbert
A. Tisserand



<http://www.lirmm.fr/arith18>

Thank you.