Simple method of computing a function in fixed-point arithmetic

S.S. Deo and K.V.S. Hari

Indexing term: Digital arithmetic

The authors present a simple method, based on the principle of successive approximation, of computing a function using a fixed-point arithmetic processor. As an example, computing the square root function is considered and the performance is compared with Newton’s method, in terms of accuracy and the number of instruction cycles required.

Introduction: The method of successive approximation (SA) is most commonly used in analogue to digital converters (ADCs). A simple method, based on this principle, for the computation of functions in fixed-point arithmetic is proposed. The effectiveness of this method in terms of accuracy, number of cycles etc.,
depends on how effectively the inverse function is computed. This letter presents the algorithm and presents the performance of the algorithm for square root computation on a fixed-point DSP processor.

**Successive approximation (SA) method for computing a function:**

Consider a B-bit fixed-point arithmetic processor and assume that the values are represented in fractional format as

\[ X = b_b b_{b-1} \ldots b_0 b_{-2} \]  

with \( b_i \) denoting the sign bit. Consider the function

\[ y = f(x) \]

which needs to be evaluated and let \( f^{-1}(y) \) denote the inverse function.

**Algorithm:**

\[ y_0 <\!\!\!< b_b = 0, b_0 = 1, b_i = 0, \ i \neq 0^*; \text{initial guess (usually mid-point)} \]

\[ y_1 <\!\!\!< \text{a constant named shift-Val-1} \]

\[ y_2 <\!\!\!< b_b = 0, b_i = 1, b_0 = 0, i \neq 1^*; \text{a constant named shift-val-2} \]

FOR (LOOP = 1 to \( B-1 \))

START LOOP

IF (\( x < f^2(y_0) \))

\[ y_b <\!\!\!< y_i \text{ OR } y_i \]

shift right \( y_i \) by 1 bit position;

shift right \( y_i \) by 1 bit position;

END IF

IF (\( x < f^2(y_2) \))

\[ y_2 <\!\!\!< \text{AND NOT } y_2 \text{ OR } y_i \]

shift right \( y_i \) by 1 bit position;

shift right \( y_i \) by 1 bit position;

END IF

IF (\( x = f^2(y_0) \))

TERMINATE LOOP;

END IF

END LOOP

\( f(x) <\!\!\!< y_0 \)

STOP

**Remarks:**

(i) The number of iterations required for computation are equal to the number of bits used for representing \( y \).

(ii) The accuracy and computational complexity of implementing the inverse function dictates the accuracy and computational complexity of the SA method of function evaluation.

**Examples: Square root computation:** Usually, the square root of a number is evaluated using the Newton Raphson method given by the iterative formula [1]:

\[ y(n) = \frac{1}{2} \left[ y(n-1) + \frac{x}{y(n-1)} \right] \]

where \( n \) denotes the \( n \)th iteration. After some iterations, \( y(n) \) approximates to \( \sqrt{x} \).

Using the SA method, it is easy to see that for square root computation, the inverse function is simply squaring, or multiplication. This can effectively be carried out by a hardware multiplier in one clock cycle, which is part of a digital signal processing (DSP) processor (e.g., [4]).

**Performance comparison:** A uniform random number generator was used to generate 100 random numbers between zero and one, and the square root was calculated for each of them using the two methods. The average error was chosen as the performance measure. Table 1 presents the results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cycles per iteration</th>
<th>Iterations</th>
<th>Average error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton-Raphson</td>
<td>32</td>
<td>40</td>
<td>( 5 \times 10^{-4} )</td>
</tr>
<tr>
<td>Successive approx</td>
<td>24</td>
<td>15</td>
<td>( 2 \times 10^{-3} )</td>
</tr>
</tbody>
</table>

**Sine inverse computation:** The proposed method was also used for computing the \( \sin^{-1} \) function in [5] and it has been shown that it performs better than the standard method using a series expansion [2].

**Conclusion:** A simple method, the SA method, is proposed for computing a function in fixed-point arithmetic. The method is general and its power lies in efficient implementation of the inverse function. As an example, it has been shown that using a DSP processor, computing the square root using the proposed method is more accurate and requires less computation compared to the conventional Newton-Raphson method.