Specialized processors and algorithms for computing standard functions

Mamirjon Turdimatov, Farrukh Mukhtarov, Sultanali Abdurakhmonov, Umidjon Khudoynazarov, and Mastura Muminova

1Fergana Branch of TUIT, Department of Information Security, 150100, Fergana, Uzbekistan

Abstract. This article discusses the problem of creating a specialized processor device that allows you to increase the number of calculated mathematical functions, resulting in a single calculation scheme for all elementary functions, an algorithm for lowering the degree of a polynomial by the Chebyshev polynomial method and a single calculation scheme for the Horner scheme, by comparing the prototype by introducing a specialized reprogramming block processor device. Keywords: microprogram, parallel, parallelization, algorithm, processor, device, memory, micro command.

1 Introduction

The goal of the task is to create a specialized processor device that allows increasing the number of calculated mathematical functions by introducing a random-access memory unit, a De multiplexer and a recording control unit into the specialized processor device, which allows reprogramming the specialized processor device with commands from the central processing unit of an electronic computer in which this specialized processing unit.

Now we will consider the calculation of power polynomials and give methods for the transition from software to hardware implementation [1].

Usually, all algorithms for calculating functions have a common identical part: they require using the appropriate formulas to bring the entire interval of variation of the argument of a given function to a certain standard interval, where the approximation is carried out.

As you know, the time spent on such a reduction is the same for all algorithms. It may turn out that one algorithm has high performance and takes up a large amount of memory, and the other has low performance with a small amount of memory. The question arises of which of them is more efficient for hardware implementation. The choice of a specific algorithm will be influenced by the goals that we strive to achieve with hardware implementation, and the technical means that we will have at our disposal.

Now we will consider the questions of a generalized algorithm for calculating elementary functions using the method of approximation by polynomials of the best uniform approximation.

As stated by the existing algorithm for calculating elementary functions [2] does not meet the requirements of a special purpose.

* Corresponding author: fer.sapedu@gmail.com
Therefore, the proposed method for calculating elementary functions \( f(x) \) - by means of their expansion into series or by their approximations by polynomials of the best uniform approximation - is based on the calculation of a polynomial of degree \( n \) from a given argument \( x \), which has constant coefficients for a given function. If a computer allocates an area in fast memory to store these coefficients, then the time for calculating elementary functions will depend on how quickly we can calculate the polynomial of the \( n \)th degree [3,4].

The tendency to reduce the cost of permanent memory and the use of only two arithmetic operations (multiplication and addition) make this method and, accordingly, the algorithm very promising for the hardware implementation of elementary functions, especially in special-purpose machines.

A function expanding into a Taylor series is more convenient to uniformly approximate a function on a segment with an accuracy of \( \varepsilon \) using the method of finding the best uniform approximation.

Now we formulate a generalized algorithm for calculating elementary functions (Fig. 1).

The operation of the algorithm begins with the selection of a segment of the Taylor series that approximates the given function with higher accuracy than the specified accuracy, i.e.

\[
f(x) \approx P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n + a_{n+1}x^{n+1}.
\]

\[
\|f(x) - P_{n+1}(x)\|_{[-\alpha, \alpha]} \leq 10^{-11},
\]

Where \( n \) is taken from table 2.1 and \( x \in [-\alpha, \alpha] \).

1. Definition \( \alpha \) for each function.

2. Make a function transformation when \( x = \alpha t \) on the segment \([-1,1]\):

\[
S_{n+1}(t) = P_{n+1}(\alpha t) = a_0 + \alpha a_1t + \alpha^2a_2t^2 + \ldots + \alpha^{n+1}a_{n+1}t^{n+1}.
\]
3. Formation of a new series using the numerical coefficients of Chebyshev polynomials

\[ \alpha^{n+1} a_{n+1} 2^{1-n} T_{n+1} \]

Where \( T_{n+1}(t) \) - Chebyshev polynomials of the first kind:
- for even and odd functions is proposed by the formula
  \[ T_{n+1}(t) = 2xT_n(t) - T_{n-1}(t); \]
- for biased functions - according to the formula
\[ T^*_n(t) = T_n(2t - 1). \]

As a result, we obtain a new type of polynomials:

\[ S_n = S_{n+1} \pm \alpha^{n+1} a_{n+1} 2^{1-n} T_{n+1} = b_0 + b_1 t + \ldots + b_n t^n. \]

4. Estimate the error of a polynomial:

\[ \| S_{n+1} - S_n \|_{[-\alpha, \alpha]} = \| \alpha^{n+1} a_{n+1} 2^{1-n} T_{n+1} \|_{[-1,1]} \leq 10^{-k} \]

Where \( \| T_{n+1} \|_{[-1,1]} \leq 1, \ \max |T_{n+1}| = 1. \)

5. Condition check

\[ \| S_m - S_{m-1} \|_{[-1,1]} \leq L_1 \cdot 10^{-10} \]

Where \( m \ll n. \)

6. Jump to the original variable \( x, \)

\[ P_{m-1}(x) = S_{m-1} \left( \frac{x}{\alpha} \right) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{m-1} x^{m-1} \]

Where \( m \ll n. \)

7. Calculate error estimates:

\[ \| f - P_{n+1} \|_{[-\alpha, \alpha]} + \| S_{n+1} - S_n \|_{[-1,1]} + \ldots + \| S_m - S_{m-1} \|_{[-1,1]} < L_2 \cdot 10^{-10} \]

8. We check the following condition

\[ |a_{n+1}| \cdot 2^{-n} - \| f - P_{n+1} \|_{[-1,1]} > \xi, \]

If the condition is met go to step 9, otherwise go to step 1.

9. Polynomial \( P_{m-1}(x), \) approximating uniformly on a segment \([-\alpha, \alpha]\) function \( f(x) \) with a given accuracy \( \varepsilon. \) Has a minimum degree.

The experiments carried out on the computer on well-known methods: “Function expansion into a Taylor series”, “Best approximation”, and “Fractional-rational approximation” show that to obtain trigonometric functions \( \sin x, \cos x, \tan x \) and \( \cotan x \) with an accuracy of 10-10 by the method the expansion of a function in a Taylor series requires 9 of the 13 terms of the series, respectively, in a certain interval, and the proposed method for finding polynomials of the best uniform approximation using a generalized algorithm gives the same accuracy with 5 terms. To calculate the inverse trigonometric functions \( \arcsin x \) and \( \arctan x, \) a minimum degree of 6 is sufficient, together with 11 terms of the Taylor series; the
The method of fractional-rational approximation requires 9 addition operations and 1 division operation.

The obtained results show that the algorithms developed by us are more efficient in comparison with other methods. For example, the results by reducing the degree of a polynomial while maintaining accuracy and the generalized algorithm allow a 2-fold increase in the calculation speed and reduce hardware costs, compared with the Taylor series expansion method and the fractional-rational approximation method.

The proposed method and algorithm is applied to special-purpose computers, processors and aircraft.

Efficiency is usually measured either by the number of costs required to obtain certain costs, an integral characteristic, or a system efficiency criterion.

From this point of view, a high-speed algorithm and device are proposed for providing specialized digital computers.

Based on the above, we will calculate the exp(x) functions according to the formula:

\[ e^x = 2^{[z]} (e^u)^2, \]

Where, \(z = x/\ln 2\); \([z]\) - closest integer to \(z\);
\(u = \{z\} \ln 2/2\), \(\{z\} = z - [z]\) fraction.

To calculate \(e^x\), we use the expansion in a Taylor series into the interval \([-1,1]\). Moreover, it is enough to take 14 terms of the series so that the absolute error does not exceed 10^{-11}, i.e., \(|R_{14}| < 10^{-11}\).

Under the influence of the method of finding the polynomial of the best uniform approximation, we lower the degree of the series. Moreover, the error after that does not exceed 10^{-10}. It can be seen that in place of 14 members remains 4.

For \(|u| \leq 0.5e^u\) calculated by the following formula:

\[ e^u = a_0 + a_1u + a_2u^2 + a_3u^3 + a_4u^4, \]

Where \(a_0, a_1, a_2, a_3,\) and \(a_4\) are the coefficients obtained by the method of the generalized algorithm for calculating functions [5,6]:

\[ a_0 = 1,000, \quad a_1 = 0,693, \quad a_2 = 0,240, \quad a_3 = 0,055, \quad a_4 = 0,009; \]

If we apply Horner's scheme, then

\[ e^u = a_0 + u(a_1 + u(a_2 + u(a_3 + a_4u)))). \]

The algorithm consists of the following:

\[ b_4 = a_4; \quad b_3 = a_3 + ub_4; \quad b_2 = a_2 + ub_3; \quad b_1 = a_1 + ub_2; \]
Software-Oriented Algorithm:
1. $C_1 = 1.44269504089$;
2. $C_2 = 0.34657354028$;
3. $C_3 = C_1 \cdot x$;
5. Overflow at $[C_3] > 128$;
6. $C_4 = [C_3]$;
7. $C_5 = C_3 - C_4$;
8. $C_6 = C_5 \cdot C_2$;
9. $C_7 = a_4$;
10. $C_8 = a_3 + C_6 \cdot C_7$;
11. $C_9 = a_2 + C_6 \cdot C_8$;
12. $C_{10} = a_1 + C_6 \cdot C_9$;
13. $C_{11} = a_0 + C_6 \cdot C_{10}$;
14. $C_{12} = C_{11}^2$;
15. $C_{13} = 2C_4$;
16. $C_{14} = C_{13} \cdot C_{12}$;
17. $e^x = C_{14}$.

It can be noted that this algorithm is easily applied to obtain arbitrary precision (on demand).

For standard exponent type ($\sin x$, $\cos x$, $\tan x$, $\cot x$, $\arcsin x$ and $\arctan x$) functions, a computing device was developed in [5].

We also propose a method of parallel computing and parallelization of the device, based on two arithmetic operations - multiplication and addition, to increase the speed of devices [7].

The disadvantage of the known device is a limited class of tasks to be solved, i.e., the ability to calculate only one function $y = \ln(1 + x)$ and slow performance.

In a known device, this function is approximated by a segment of the Taylor series, which is calculated according to the Horner scheme:

$$F(x) = b_0 + \Delta x(b_1 + \Delta x(b_2 + \Delta x(b_3 + \Delta x(b_4 + 0 \cdot \Delta x)))),$$

(1)

Where $F(x)$ - computed function;
$\Delta x$ - the difference between the argument values and the nearest integer;
$b_i$ - nodal points

The execution of actions according to formula (1) to find the value occurs sequentially, i.e., the product is calculated first $0 \cdot \Delta x$ then the sum $0 \cdot \Delta x + b_4$ etc. (those is an iterative process). Therefore, the calculation time of the function at work [1].

$T^{c.a.} = 5(t_{mul} + t_{addit} + t_{acc})$ or in general for $m$ -iterations:

$T^{c.a.} = m(t_{mul} + t_{addit} + t_{acc})$,

Where $m$ - the highest degree of the polynomial calculated by Horner's scheme;
$t_{mul}$ - multiplication operation time;
$t_{addit}$ - additional operation time;
\( t_{\text{acc}} \) - ROM access time.

We propose a device that performs actions according to formula (1) to find the value that occurs sequentially, i.e., the product is calculated first \( 0 \cdot \Delta x \) then sum \( 0 \cdot \Delta x + b_4 \) etc. (those is an iterative process). Therefore, the time for calculating the function for our device without selection is the time for sampling from the ROM.

The proposed device is more advanced - expanding the class of tasks to be solved due to the possibility of calculating all the functions that lead to the solution according to the Horner scheme and increasing the speed by combining the operations of multiplication and fetching from ROM.

This goal is also achieved by the fact that a third register and a bus buffer are additionally introduced into the device, the first port of which is connected to the computer, and the second port is connected to the inputs of the first, second and third registers, as well as to the outputs of the first and second groups of gates having three states, the second group of inputs of the adder is connected to the outputs of the third register, and the first and second groups of inputs of the multiplier are connected respectively to the outputs of the first and second registers, the outputs of the adder are connected to the inputs of the first group of gates, the corresponding outputs of the control unit are connected to the control inputs of the third register, bus buffer and ROM. Figure 2 shows a block diagram of the proposed device.

![Block diagram of the proposed device](image)

**Fig. 2.** Device for calculating functions.
Fig. 3. Timing diagram of the operation of the device for calculating functions.

The circuit includes registers 1, 2 and 3, a multiplier 4, an adder 5, the first 6 and second 7 gate groups, ROM 8, a bus buffer 9, a control unit 10, the first group of inputs of the multiplier 4 is connected to the outputs of register 1, and the second the group of inputs is connected to the outputs of register 2, the first group of inputs of the adder 5 is connected to the outputs of the multiplying device, and the second group of inputs is connected to the outputs of register 3, the inputs of the first group of gates 6 are connected to the outputs of the adder, the inputs of the second group of gates 7 are connected to the outputs of the ROM 8, the first port of the bus buffer 9 the first port is connected to the inputs 1, 2, and 3 of the registers and the outputs of the first and second groups of gates, and the second port of the bus buffer is connected to the data bus of the computer, with the control inputs of the registers, ROM gates and the bus buffer are connected to the corresponding inputs control unit 10.

In the initial state, gates 6 and 7, as well as bus buffer 9, are in the third state, in addition, bus buffer 9 is configured to receive information from the computer.

The creation of specialized computing devices designed to solve special problems is accompanied by well-known difficulties associated with the need to increase their speed and calculation accuracy.

Methods for constructing such devices are based on the implementation of new calculation methods, improvement and wider use of a complex of independent computing units.

A structural scheme for organizing the calculation of elementary functions is proposed, consisting of a device that converts numbers from a generalized interval to a standard one and vice versa. A device has been developed that converts floating-point numbers to fixed-point numbers and vice versa.

To improve the technical characteristics of specialized devices, in the conditions of widespread use of large integrated circuits and the development of control systems for a digital computer, a structural organization of the calculation of elementary functions is proposed to ensure the high accuracy of the system. Computer control systems and their development largely depend on the available central processor and the ability to connect additional external devices, taking into account the possibility of organizing a backbone architecture.
Studies have shown that the proposed method for calculating elementary and special functions has a single architectural solution for calculating all functions. Ensuring the homogeneity of the structure of the developed device is satisfied with a decrease in the number of operations to be calculated to recurrent formulas:

\[
F_i(x) = a_{i0} + x^2(a_{i1} + \ldots + x^2(a_{im-1} + a_{im}x^2))
\]

Where \( i = 1,2,\ldots, n \) — the number of elementary and special functions;
\( m \) — corresponding degree polynomials;
\( a_{im} \) - polynomial coefficients.

The coefficients of elementary functions are determined based on the proposed generalized algorithm for calculating elementary functions. These coefficients are the basis of the ROM of a specialized device.

2 Conclusion

- Specialized device for calculating elementary functions, the proposed device - expanding the class of tasks to be solved due to the possibility of calculating all elementary functions that are reduced to a solution according to the generalized Horner scheme and increasing speed by combining multiplication and fetching from ROM.
- The developed device is used as a central processing unit of a PC or as a peripheral device of universal computers.
- Evaluated the efficiency of the device: it improves system performance by 7-9 times compared to the existing one.
- Developed machine-oriented algorithms for calculating elementary functions. Based on the generalized algorithm, the coefficients of elementary functions in the standard interval are determined with an accuracy of 10-10. Transitional formulas from the generalized interval to the standard one and vice versa are proposed. Estimates of the error of elementary functions for different word lengths of their representation are given.
- The method of transition from the software implementation of the calculation of elementary functions to the hardware implementation is given.
- To calculate elementary functions, a parallel computing device is proposed, based on only two arithmetic operations - multiplication and addition.
- The obtained results of the research were implemented at the enterprise p / box 1-5885 in the development of on-board digital computers of the Salyut type, while the accuracy of calculating elementary functions reached 10-6 -10-11, and the calculation speed increased by 20-25% due to a decrease in the number of operations and elimination of division operations.
- The implementation of the results of the work and the main scientific provisions will not only save time for the tasks being solved but also reduce hardware costs and financial resources.

References