Optimisation and Implementation of the Arctan Function for the Power Domain

A.Th. Schwarzbacher[2,3],A. Brasching[1,2], Th.H. Wahl[1,2], P.A. Comiskey[2] and J.B. Foley[3]
[1]Fachhochschule der Deutschen Telekom AG, Dieburg, Germany
[2]Dublin Institute of Technology, Dublin, Ireland
[3]Trinity College, Dublin, Ireland

Abstract: Trigonometric functions are used in many applications including real time digital signal processing (DSP), navigation and astronomy. Today's demand for fast, small and portable equipment in those areas has resulted in the need for optimised structures for real-time processing of complex mathematical functions. Therefore, the multipurpose algorithms used traditionally for such implementations are investigated in this paper and new directions for future implementations are presented.

Keywords: CORDIC Algorithm, Low-Power Design, High-Level CMOS Design.

1 Introduction

The COrdinate Rotation Digital Computer (CORDIC) Algorithm [1] is traditionally used for the implementation of trigonometric functions. Volder first introduced the CORDIC Algorithm in 1959. Since then it has been the most popular algorithm for implementing mathematical functions. With this algorithm only shift steps and addition operations are required to calculate most mathematical functions. The basic idea of CORDIC is to take an angle and "rotate" a vector over this angle towards zero. The CORDIC Algorithm of Volder uses three input variables (x, y, z) and is based upon the algorithm shown in Figure 1.

$$x_{n+1} = x_n + d_n y_n 2^{-n}$$
$$y_{n+1} = y_n - d_n x_n 2^{-n}$$
$$z_{n+1} = z_n + d_n \arctan 2^{-n}$$

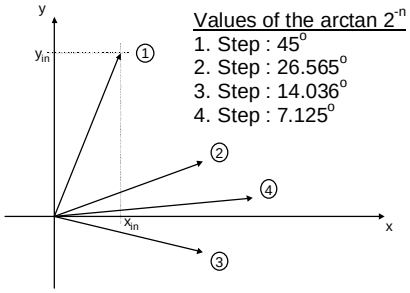


Figure 1: The Rotation Steps of the CORDIC Algorithm

The terms for *arctan* 2^{-n} are precomputed and stored, n is the index of the iteration (1, 2, 3,..., n) and the value of d_n is either +1 or -1. The variables are initialised. Next, a set of iterative equations is repeatedly applied to these variables until the result converges to the required accuracy. The accuracy can be controlled using the number of computing steps. If accuracy necessary is not the main concern, the computation can be stopped after a few steps. The different functions are selected by the value of d_n. The term d_n is chosen in such a way, that with each step either y or z is driven toward zero. Figure 1 shows the rotation steps when, that y is driven toward zero.

2 Implementation of the Arctan

To show the performance of the CORDIC Algorithm, the trigonometric function arctan was implemented in hardware. Additionally, three further solutions were developed and compared with respect to error deviation, timing behaviour, power consumption and area requirements. One implementation for the arctan uses the CORDIC Algorithm. Two other implementations use a Lookup Table and the last one uses an approximation technique. The system input has a bitwidth of 7 bits and the output is 6 bits wide. In view to the fact that the arctan function is an odd function, the sign of the input value is cut off. The sign can be assigned directly from the input to the output, because of (1)

$$arctan\ (x) = -\ arctan\ (-x). \qquad (1)$$

The input range from the arctan is from $-\infty$ to $+\infty$. The implementation of such a function is only possible by restricting the input range. The input range is restricted from 0 to 1.984375 with a bit step value of 0.015625. The output is defined such that 60° corresponds to an output 42 decimal. This results in a resolution of 1.4286° at the output. The input is normalised so that an input value of 1 corresponds to an output of 60°. Therefore, the input value is multiplied by the factor $\sqrt{3}$. All models are implemented as synchronous systems.

2.1 Using the CORDIC Algorithm

This model uses the CORDIC Algorithm as described in the previous section. Here, the value of y corresponds to the input value for the first stage and the value of z in the last stage corresponds to the output value. The design consists of 10 pipeline stages as shown in Figure 2. The required shift, addition and subtract operations are done in each stage, except the first one were no shift operation is needed. The fixed angle is stored separately in the calculation stage and will be added or subtracted from *z* depending on the value of *y*.

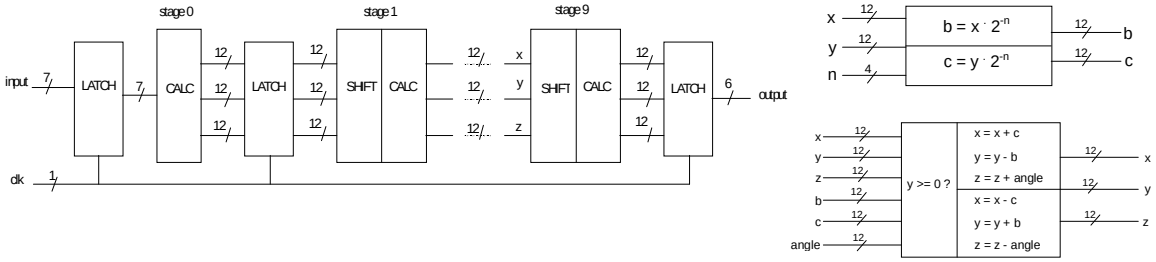


Figure 2: The Structure of the implemented CORDIC Algorithm

The result of y after several calculation steps results in *result = arctan (y/x)*. Therefore, the value of x must be set to $1/\sqrt{3}$ to achieve the actual result of *arctan* $(\sqrt{3} \cdot y)$.

2.2 Using a Lookup Table

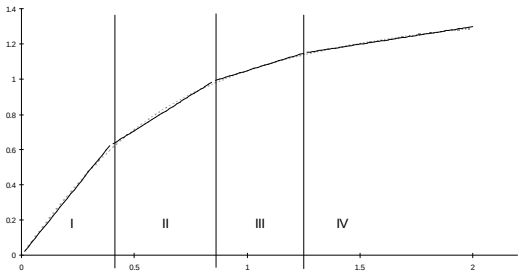
A Lookup Table (LUT) is a simple storage device. All output values are precomputed and stored in the LUT. The input is an address in the LUT, which is used to access the output data. There are no calculations. Therefore, the design is very fast. All output values are so as to minimise the error. Thus, the error deviation can not exceed 0.7143° (half of the resolution). The LUT contains 128 values, because there are 7 bits at the input (2^7=128).

2.3 Modifying the LUT

The *modified* LUT uses the same precalculated and stored data as the LUT, except for the first 23 input values, which will be directly assigned to the output. The reason for this assignments is that for small *x*, the *arctan(x)* is approximately *x*. This direct assignment of output to input is implemented by changing the resolution of both input and the output. The change in resolution at the output with the definition, that 60° responds to 42 decimal results in a factor of 1.59. By multiplying the now adjusted and normalised input value by √3, the approximation of *arctan (x) = x* is valid again. Therefore, the value of the input will be assigned direct to the output for input values from 0000000 to 0010110.

2.4 Approximating the Arctan

The fours implementation is based on an linear approximation of the arctan, which is optimised for hardware implementations. The characteristic of the arctan is divided in four sections as shown in Figure 3. Part I is based on the principle of direct assignment as used in the modified LUT before. Parts II to IV are use simple equations, which represent the desired values. All equations use a multiplier, which is a multiple of 2^{-n} and one constant, which will be added. Therefore, it is easy to implement this equation into hardware by using shift and an adding operations. The equations shown in Figure 3 are only optimised for this particular input width and the function arctan ($\sqrt{3} \cdot x$) with the condition that 42 decimal is responding to 60°.



I.	0 - 0.39063	$\arctan(\sqrt{3} \cdot x) = x$
II.	0.40625 – 0.84375	$\arctan(\sqrt{3} \cdot x) = 2^{1} \cdot x + 0.31$
III.	0.84938 – 1.25	$\arctan(\sqrt{3} \cdot x) = 2^{-2} \cdot x + 0.65$
IV.	1.26563 – 1.984375	$\arctan(\sqrt{3} \cdot x) = 2^{-3} \cdot x + 0.9$

Figure 3: The four Sections for the Approximations of the Arctan

3 Results

In this section the four different implementations of the arctan algorithm are compared. The designs were written in VHDL, synthesised using Synopsys Design Compiler without any constraints using an ES2 0.7µm technology. All implementations were designed to have a deviation of less than one bit from the theoretical value. All designs have a propagation delay of less than 10ns. In respect of the timing behaviour there is nearly no difference between the version using the Approximations and both Lookup Table versions. However, the Approximations technique uses 80% less time for the calculation than the CORDIC Algorithm.

Figure 4 shows the power consumption of the different implementations. The power consumption was established using PowerCount [2] for an operating frequency of 10MHz and a supply voltage of 5V. As can be seen in Figure 4, the Approximation technique uses only 0.6mW at 10Mhz, compared to the CORDIC Algorithm which uses 17mW a reduction in power consumption by a factor of 25. The modified Lookup Table requires 78.5% of the power, that is consumed by the normal Lookup Table. The reason for this is that the first 23 of the 128 input values (from 0000000 to 0010110) are directly assigned to the output. The version using the Approximation technique

needs only 66.7% of the power of the Lookup Table and therefore, this version is the best with respect to power consumption.

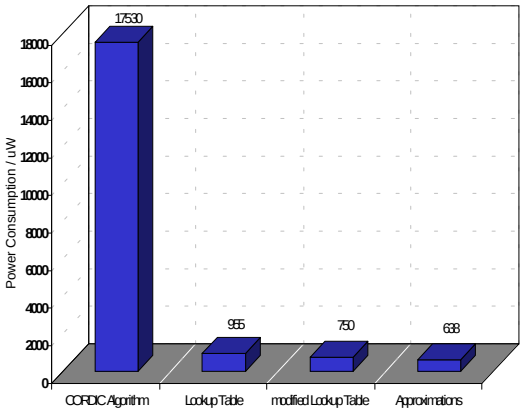


Figure 4: Power Consumption

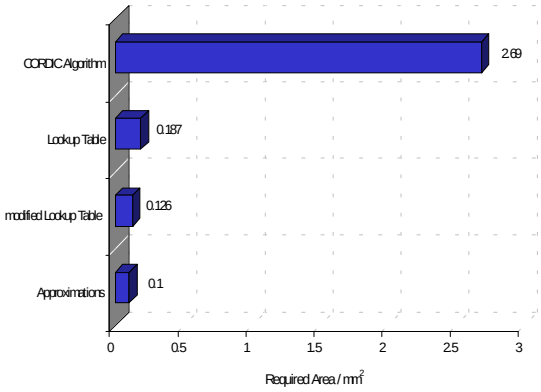


Figure 5: Area Requirements

Figure 5 presents the area requirements of the different implementations. Like the power consumption, the best result with respect to the required area is produced by the approximation technique. This version uses 53.5% of the area required by the Lookup Table, 79.4% of the area required by the modified Lookup Table and 24 times less area when compared to the original CORDIC algorithm.

4 Conclusion

The aim of this paper was to show that the investigation of traditional algorithms can reduce power consumption significantly. For this purpose the arctan function was investigated and three alternative implementations to the traditional CORDIC algorithm where implemented.

It was possible to reduce the power consumption of the traditional implementation by a factor of 25. This reduced power consumption was achieved without compromising any other performance feature such as accuracy or throughput. In fact, most other performance parameters also improved. For example the required silicon area was also reduced by a factor of 24. Therefore, the authors have shown, that traditional multi-purpose algorithms may not be optimised towards power consumption. In addition, there are many possibilities of performing the same operations with significantly reduced power consumption, without compromising the overall performance of any aspect of the implementation. Therefore it was shown that it is worth wile to invest time and resources to investigate alternative implementations of traditional algorithms.

References

[1] J.E. Volder, “The CORDIC trigonometric computing technique,” *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330-334, Sept. 1959

[2] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, “Powercount: measuring the power at the VHDL netlist level,” *Electronic Devices and Systems Conference*, Bruno, Czech Republic, pp. 70-73, June 1998.

Contact: schwarzbacher@gmx.net