Efficient Double-Precision Cosine Generation *

Derek Nowrouzezahrai Brian Decker William Bishop
dnowrouz@uwaterloo.ca bjdecker@uwaterloo.ca wdbishop@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1.
Tel: 519-888-4567 ext. 7159
Fax: 519-746-3077
Contact Author: Dr. William D. Bishop

Abstract

The trigonometric function of $\cos(\theta)$ plays an important role in communication systems, digital signal processing systems, and graphical systems. This paper presents a technique for generating the cosine of an angle that is more computationally efficient than the CORDIC algorithm and its many variants for double-precision floating point calculations. A hardware design implementation of the cosine generator has been developed. Simulation results for an Altera Stratix II FPGA implementation indicate that the hardware design is both more efficient and more precise than previous published implementations.

Keywords:

Cosine generation, computer arithmetic, FPGA, CORDIC, IEEE floating point

1. Introduction

The calculation of cosine functions is crucial in digital signal processing, communications and graphical systems. With the ever increasing complexity of these systems and the increasing demands on data rates and quality of service, efficient calculation of the cosine function with a high degree of accuracy is vital. The most widely implemented algorithm used for calculation of the cosine function is the CORDIC algorithm introduced in 1959 by Volder[4]. The CORDIC algorithm is based on simple adds and shifts to perform multiplication and division. This makes the hardware demands for such a cosine calculator rather minimal, except when high precision is required. Consequently, current implementations of cosine calculators based on the CORDIC algorithm offer single-precision results. This paper proposes an implementation of a double-precision cosine generator based on a Taylor Series expansion of the cosine function as follows:

$$\cos(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{2n}}{2n!}, \text{ for all } \theta \quad (1)$$

Given an input angle θ and using only the first eleven coefficients of the series, it is possible to calculate a double-precision answer for the cosine function[5]. The coefficients can be calculated and hardwired into registers for use in the cosine calculation.

2. Algorithms for Cosine Generation

The implementations of cosine calculation algorithms is analyzed in this section. The CORDIC algorithms in rotation mode with and without redundant number systems [3] are looked at, in addition to the algorithm proposed in this paper.

2.1. CORDIC Algorithm

The CORDIC algorithm for calculation is based on the following iteration:

$$x_{n+1} = x_n - a_n y_n 2^{-n} \quad (2)$$
$$y_{n+1} = y_n + a_n x_n 2^{-n} \quad (3)$$
$$z_{n=1} = z_n - a_n \arctan 2^{-n} \quad (4)$$

* This research was supported by Altera Corporation.

The $\arctan 2^{-n}$ values are precomputed and stored. In the rotation mode of CORDIC, a_n is chosen as follows:

$$\begin{aligned} a_n &= +1, \text{ when } z_n \geq 0 \\ a_n &= -1, \text{ when } z_n < 0 \end{aligned}$$

If $|z_0| \leq \Sigma_{n=0}^{\infty} \arctan 2^{-n}$, then:

$$\begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \text{ as } n_{\rightarrow\infty} K \begin{pmatrix} x_0\cos(z_0) - y_0\sin(z_0) \\ x_0\sin(z_0) + y_0\cos(z_0) \\ 0 \end{pmatrix}$$

where K is equal to $\prod_{n=0}^{\infty} \frac{1}{\cos(a_n \arctan(2^{-n}))}$ [3].

Thus to compute the sine or cosine of an angle, one can choose $z_0 = \theta, y_0 = 0$ and $x_0 = \frac{1}{K}$. To achieve double precision results, iteratively computing the CORDIC algorithm usually requires 1 cycle for each bit of precision. Thus, to achieve precision equal to that of IEEE double-precision floating point numbers for $0.1 \leq \cos(\theta) \leq 1.0$, 52 iterations would be required, as this is the number of mantissa bits maintained in this structure. However, for values less than 0.1, more iterations are required to achieve the precision of IEEE double-precision floating point. Storing the input vectors $x_0,\ y_0,\ z_0$ in IEEE double-precision floating point representation would defeat the purpose of the CORDIC algorithm as floating point additions would be required and the bit shift operation would require a floating point multiply. For this reason, the input vectors are often stored in simple binary representation. Consequently, additions for each iteration are performed in a ripple-carry format. As a result, increasing the precision required for these additions increases the total delay of the adder. This can be alleviated, however, by storing the iterations in a binary signed-digit representation. This is known as the CORDIC implementation with Redundant Number Systems [3].

The overall delay of an adder designed for the binary signed-digit number representation is static with respect to bit resolution. However, the complexity of the adder for inputs in binary signed-digit representation is 4-fold that of a ripple-carry adder for the same bit resolution. In addition to this, the determination of the sign of z_n for the determination of a_n is not as easy since a single sign bit cannot be checked. Summarizing, the drawbacks to the CORDIC algorithm are as follows:

- Precision equivalent to IEEE double-precision floating point requires significantly more iterations for values closer to zero
- a_n is dependent on the sign of z_n, and depending on the representation of z_n this step can potentially be difficult
- Increasing precision of the CORDIC algorithm requires increased complexity of bit shift and addition hardware
- Increased precision requires larger lookup table for precomputed $\arctan 2^{-n}$ terms

2.2. Proposed Algorithm

As mentioned previously, the Taylor Series for a cosine function is as follows:

$$\cos(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{2n}}{2n!}, \text{ for all } \theta \tag{5}$$

However, for calculations of cosine values we will restrict the range to $-2\pi \leq \theta \leq 2\pi$. For standard IEEE double-precision accuracy 64 bits are used to represent the floating point number; 52 mantissa bits, 11 exponent bits, and 1 sign bit. To allow for proper rounding, the cosine function should be calculated to at least 55 mantissa bits. Therefore, the bit in the last position of the mantissa carries the following weight:

$$K = 2^{-55} \approx 2.78 * 10^{-17} \tag{6}$$

Consequently, it is reasonable to evaluate the Taylor Series terms until the magnitude of the final term is less than half that of the bit in the last position. By trial and error, it is revealed that:

$$\frac{1}{2}K \geq \frac{\theta}{(2*10)!} \text{ for } |\theta| < 33.76 \tag{7}$$

This restriction on θ is well within the range $-2\pi \leq \theta \leq 2\pi$ and consequently it is obvious that the use of 11 terms of the Taylor Series expansion is sufficient to compute $\cos(\theta)$ to double-precision accuracy. The Taylor Series expansion for double precision then reduces to:

$$\cos(\theta) = \sum_{n=0}^{10} \frac{(-1)^n \theta^{2n}}{2n!} \text{ for } -2\pi \leq \theta \leq 2\pi \tag{8}$$

2.3. Comparison of Algorithms

For the CORDIC algorithm the total computational requirements for x_{n+1} and y_{n+1} in each iteration is as follows:

- 2 binary bit shifts
- 1 binary addition
- 2 binary subtractions

Assuming for the moment that $0.1 \leq \cos(\theta) \leq 1.0$ for precision equivalent to IEEE double-precision floating point, the following computations are required:

- 104 binary bit shifts
- 52 binary additions

- 104 binary subtractions

In the proposed Taylor Series expansion, a single cosine calculation requires the following operations:

- 20 floating point multiplies
- 10 floating point additions

3. Cosine Generator Design

The cosine generator design precomputes Taylor Series coefficients for the first 11 elements of the series. This reduces the computation requirements for calculating the cosine function down to simple add and multiply routines. Precomputing the coefficients eliminates the need for the division required which is costly in terms of zperformance and hardware requirements. Using this principle the hardware implementation serves to compute $\cos(\theta)$ in the following manner:

$$\cos(\theta) = 1 - C_0 * \theta^2 + C_1 * \theta^4 - C_2 * \theta^6 + \cdots - C_9 * \theta^{20} \quad (9)$$

For double-precision floating-point calculations, it is necessary to calculate the coefficients C_i accurate to 19 decimal places. The following coefficients are required:

$$\begin{aligned}
C_0 &= 0.5000000000000000000 \\
C_1 &= 0.0416666666666666667 \\
C_2 &= 0.0013888888888888889 \\
C_3 &= 0.0000248015873015873 \\
C_4 &= 0.0000002755731922398 \\
C_5 &= 0.0000000020876756987 \\
C_6 &= 0.0000000000114707455 \\
C_7 &= 0.0000000000000477947 \\
C_8 &= 0.0000000000000001561 \\
C_9 &= 0.0000000000000000004
\end{aligned}$$

This model forms the basis for the hardware design of the cosine generator.

3.1. Structure

Figure 1 highlights the structural design of the hardware cosine implementation.

As illustrated by the figure, six double-precision, floating point multipliers, and three double-precision, floating point adders are needed to complete each stage of the cosine calculation. However, the Altera floating point multiplier megafunction chosen for implementation has a five-stage pipeline compared with a single stage pipeline for the design the floating point adder [1]. This allows the addition to catch up with the multipliers in the final stages of calculation resulting in the use of only a single adder. A negative effect to the overall design is that the multiplier pipeline cannot be fully utilized since each stage of the computation requires the multiplier outputs θ^{2n} from the previous stage.

Total ALUTs	7634/48352	15%
Total Registers	4516	-
DSP Block 9-bit Elements	156/288	54%

Table 1. Hardware Implementation Synthesis Results

3.2. Hardware Synthesis

The hardware design was synthesized using Altera Quartus Version 4.2 for a Stratix II device[2]. The results of this design can be seen in Table 1. Each of the floating-point multiplier requires 26 of the Stratix IIs 9-bit DSP elements, accounting for the 156 required in this design.

3.3. Simulation Results

When the design was simulated, it was found that the maximum clock frequency was 69.24 Mhz. This is reasonable the complexity required to implement floating point adders and multipliers. As a result of the 5 clock cycle latency of the multipliers, the overall design can generate a result in a total of 42 clock cycles. This results in each cosine calculation being computed in 606.6 ns with an overall throughput of 1.649 MS/s for all ranges of θ. Since many of the coefficients underflow the IEEE double-precision floating point multiplier, for certain restrictions on θ they can be considered to be zero. Figure 2 shows simulation results based on a range for θ, specifically which stage in the calculation the result can be pulled from to achieve double-precision accuracy. Table 2 illustrates the throughput of cosine calculations for the ranges on θ.

4. Conclusions and Future Work

We have managed to create a cosine generator which is able to produce double-precision results with very high throughput. However, the current implementation can be further optimized through the design of a single cycle double-precision floating point multiplier. A design in this manner would significantly reduce the number of clock cycles to complete each cosine calculation. How-

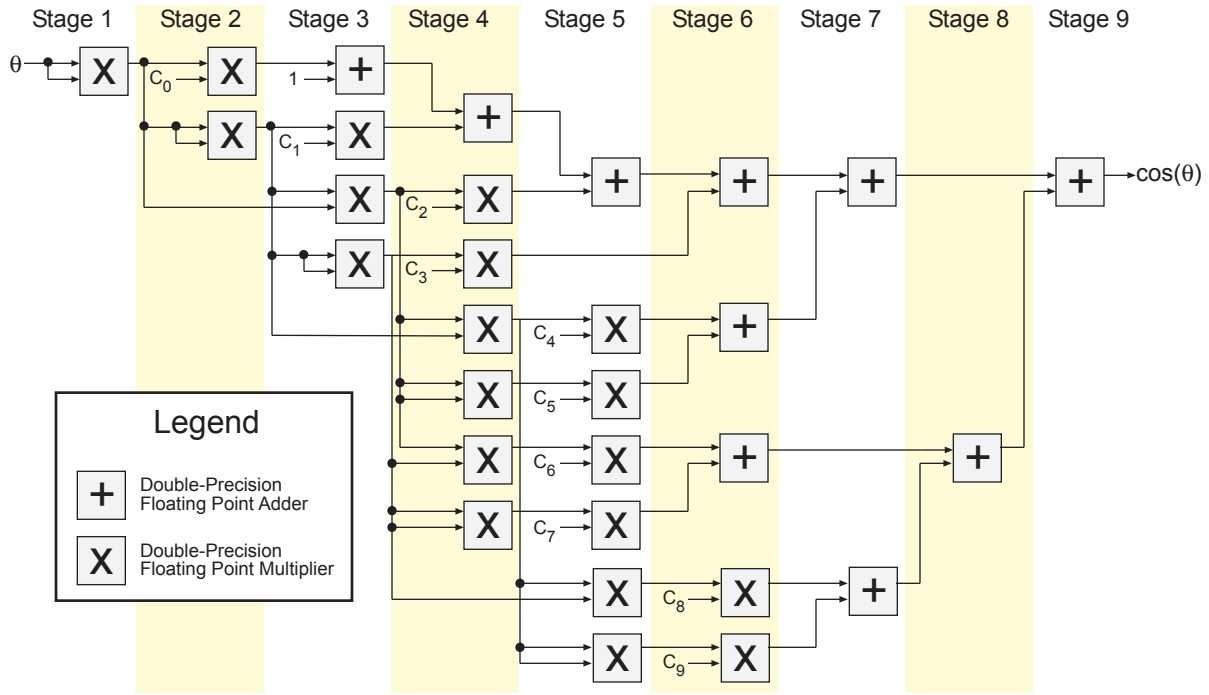

Figure 1. Cosine Generator Block Diagram

θ Range	Number of Coefficients	Clock Cycles	Maximum Performance
0 - 2	3	25	2.77 MS/s
3 - 7	4	26	2.66 MS/s
8 - 16	5	31	2.23 MS/s
17 - 27	6	32	2.16 MS/s
28 - 43	7	33	2.10 MS/s
44 - 61	8	34	2.04 MS/s
62 - 81	9	41	1.69 MS/s
82 - 90	10	42	1.65 MS/s

Table 2. Cosine Generator Throughput

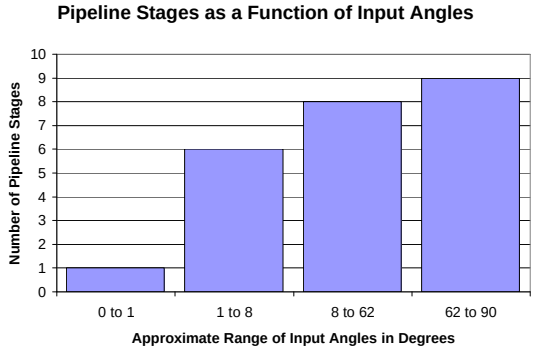

Figure 2. Number of Required Stages in the Pipeline

ever, it would be expected that a degradation in maximum clock speed would be experienced.

References

[1] Altera Corp. Floating-Point Multiplier. Functional Specifications A-FS-04-01, Altera Corp., San Jose, California, Jan 1996.

[2] Altera Corp. Stratix II Device Overview. Data Sheet DS-STXGX-2.2, Altera Corp., San Jose, California, Dec 2004.

[3] J. Duprat and J. Muller. The CORDIC algorithm: New results for fast VLSI implementation. *IEEE Transactions on Computers*, 42(2):168–178, Feb. 1993.

[4] J. Volder. The CORDIC computing technique. *IRE Trans-*

actions on Electronic Computers, EC-8(3):330–334, 1959. Reprinted in E. E. Swartzlander, *Computer Arithmetic*, Vol. 1, IEEE Computer Society Press Tutorial, Los Alamitos, CA, 1990.

[5] W. Bishop. Design of a Low Power High Performance Cosine Generator. Technical report, University of Waterloo, Waterloo, Canada, N2L 3G1, Dec 1994.