Hardware Operator for Simultaneous Sine and Cosine Evaluation

Arnaud Tisserand

HARDWARE OPERATOR FOR SIMULTANEOUS SINE AND COSINE EVALUATION

Arnaud Tisserand

LIRMM, CNRS–Univ. Montpellier II
161 rue Ada. 34392 Montpellier cedex 5. France
arnaud.tisserand@lirmm.fr

ABSTRACT

This work deals with hardware evaluation of the sine and cosine functions for the same argument simultaneously. The proposed method uses trigonometric identities, small lookup tables and low-degree polynomial approximations with very sparse coefficients. Most of the multiplications are replaced by a small number of additions or subtractions, this leads to small and fast circuits.

1. INTRODUCTION

High-speed approximations to the sine and cosine functions are often used in digital signal and image processing or in digital control. Several methods have been proposed to evaluate those functions, see [1] for a survey. When the input/output precision is relatively low (less than 24 bits), table and addition methods are often employed [2, 3]. Efficient methods based on small multipliers and tables have been proposed [4].

In this work, we show that using some trigonometric identities and simple basic operations, it is possible to evaluate both functions in about the same time as to compute only one of them and with small circuit. The proposed method is based on small lookup tables and low-degree polynomial approximations with sparse coefficients. The complete operator requires at most one small multiplication, all other multiplications have been transformed into a small number of additions/subtractions. Our method is implemented using Maple.

2. PRELIMINARIES

The proposed method uses the following trigonometric identities. These identities are widely used in software evaluation of trigonometric functions, for instance see [5].

$$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y), \quad (1)$$
$$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y). \quad (2)$$

The notation $()_2$ denotes the binary representation of a value (e.g., $3.125 = (11.001)_2$). The quantified values will be represented in the *borrow-save* format [6] (i.e., radix-2 redundant representation with the digit set $\{-1, 0, 1\}$. Bits with a negative weight are denoted by $\overline{1}$.

The approximation error ϵ_{th} due to the use of polynomial P to evaluate function f on $[a, b]$ is defined below and numerically estimated using the Maple `infnorm` function:

$$\epsilon_{th} = ||f - P||_\infty = \max_{a \le x \le b} |f(x) - P(x)|.$$

The input argument x is considered as exact. The *approximation error* measures the distance between the mathematical function and the approximated function used to evaluate it. The *rounding error* due to the discrete nature of the final and intermediate values adds up to the approximation error.

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. Minimax approximations can be computed thanks to an algorithm due to Remes [7] (numerically computed using the Maple `minimax` function).

As we deal with hardware implementations, multiplications by powers of 2 reduce to shifts (only routing).

3. PROPOSED ALGORITHM

We propose a method to evaluate simultaneously $\sin(x)$ and $\cos(x)$ with the argument x in the domain $[0, \pi/4]$. Efficient range reduction methods can be used to reduce to that domain [1]. However, our method can be applied to other domains. The argument x is a w_I-bit number and the outputs $\sin(x)$ and $\cos(x)$ are w_O-bit numbers (w_I and w_O are parameters provided by the user).

The proposed method is based on the following steps:

Step 1: argument decomposition $x = a + h$;

Step 2: parallel evaluation of

- $\sin(h)$ and $\cos(h)$ by polynomial approximation;
- $\sin(a)$ and $\cos(a)$ by table lookup;

Step 3: results reconstruction using

$$\sin(a+h) = \sin(a)\cos(h) + \cos(a)\sin(h); \quad (3)$$
$$\cos(a+h) = \cos(a)\cos(h) - \sin(a)\sin(h). \quad (4)$$

3.1. Argument Decomposition

The argument x is decomposed into $x = a + h$ with

- a the m most significant bits (MSB) of x;
- h the $l = w_I - m$ least significant bits (LSB) of x.

Here we restrict h to positive values in order to avoid sign extension problems and to simplify the decomposition process (only routing at the hardware level). The parameter m will be used for tuning speed/accuracy tradeoffs.

3.2. Polynomial Approximations to $\sin(h)$ and $\cos(h)$

We use minimax polynomials as a starting point. The coefficients of the minimax approximations will be modified to sparse coefficients in a second time. Minimax approximations provide better accuracy than Taylor series expansions because there are not only accurate around one point [1].

As the value h is small, i.e., $h < 2^{-m}$, we can use low-degree polynomials to approximate $\sin(h)$ and $\cos(h)$. In the following, we will consider degree-d polynomial approximations with $d \in \{1, 2\}$ (d is a parameter). $P = p_0 + p_1 h + p_2 h^2$ denotes the polynomial that approximates $\sin(h)$. $Q = q_0 + q_1 h + q_2 h^2$ is the polynomial approximation to $\cos(h)$.

The hardware cost for polynomial approximation is dominated by multipliers. In order to avoid the products of the coefficients by the powers of h, we use quantified coefficients with at most k non-zero bits and $k << w_O$. This means that p_1, p_2, q_1 and q_2 have at most k non-zero bits (1 or -1). The quantification is performed on the minimax polynomial coefficients using a Maple program for the parameter k. The parameter k is determined using a very simple greedy algorithm. We start with $k = 2$ and increment it at each iteration. We stop when the target accuracy is obtained or the maximal value k_{max} is reached.

The P and Q values are expressed using the notation below where the bits $p_{i,j}$ and $q_{i,j}$ are in $\{-1, 0, 1\}$, and the exponents $\gamma_{i,j}$ and $\delta_{i,j}$ are natural integers less than w_O.

$$P = p_0 \; + \; \sum_{i=1}^{d} \underbrace{\sum_{j=1}^{k} p_{i,j} 2^{\gamma_{i,j}}}_{p_i} \; \times h^i,$$

$$Q = q_0 \; + \; \sum_{i=1}^{d} \underbrace{\sum_{j=1}^{k} q_{i,j} 2^{\delta_{i,j}}}_{q_i} \; \times h^i.$$

For instance with $d = 2$, $k = 3$, $w_O = 16$ and $h < 2^{-4}$ ($m = 4$), the minimax polynomial approximation to $\sin(h)$ is $P = 1.000366h - 0.015620h^2$. The coefficient p_1 is quantified to $(1.000000000011)_2$ and p_2 to $(0.000001)_2$. The evaluation of P requires 4 additions/subtractions and one multiplication to produce h^2 (h is only l-bit large). This polynomial leads to an approximation with 18.5 bits of accuracy. This is more accurate than the order-3 Taylor series expansion of $\sin(h)$, which is $x + O(x^3)$, that only provides an approximation with 14.5 bits of accuracy.

3.3. Lookup Tables for $\sin(a)$ and $\cos(a)$

TS denotes the table that stores $\sin(a)$ values. TC is the table for $\cos(a)$ values. Both tables have 2^m entries.

In order to replace multiplications by a small number of additions in the results reconstruction, the values stored in the tables are quantified to borrow-save values with at most t non-zero bits with $t << w_O$ (t is another parameter). When using quantified values with t non-zero bits, the product $\sin(a) \times P$ (or $\cos(a) \times Q$) is replaced by $t - 1$ additions or subtractions.

The TS and TC values are expressed using the notation below where the bits s_i and c_i are in $\{-1, 0, 1\}$, and the exponents α_i and β_i are natural integers less than w_O.

$$TS(a) = \sum_{i=1}^{t} s_i 2^{\alpha_i} \quad \text{and} \quad TC(a) = \sum_{i=1}^{t} c_i 2^{\beta_i}.$$

The quantification error for each table is defined by:

$$\epsilon_S = \max_a |TS(a) - \sin(a)|;$$
$$\epsilon_C = \max_a |TC(a) - \cos(a)|.$$

For instance the quantification of $\sin(\pi/8) \approx 0.382683$ with $t = 2$ gives the binary value $v = (0.011)_2$. The quantification error is $|v - \sin(\pi/8)| \approx 0.007683$ which corresponds to 7.02 bits of accuracy.

In practice, the values in TS and TC are stored using a specific representation in order to simplify the reconstruction. Using a direct sparse borrow-save representation requires to get the position of the t non-zero bits (using a decoder). The values stored in the tables are couples (position, weight) of a non-zero bit where position is an integer less than w_O and weight is in $\{-1, 0, 1\}$. This avoids costly decoders.

3.4. Results Reconstruction

The reconstruction of the two results $\sin(x)$ and $\cos(x)$ is performed by developing all the products with respect to quantified polynomial coefficients and quantified table values in the identities:

$$\sin(a + h) = \sin(a)\cos(h) + \cos(a)\sin(h);$$
$$\cos(a + h) = \cos(a)\cos(h) - \sin(a)\sin(h).$$

The product $\sin(a) \times \cos(h)$ (or one of the three other products) is performed by using the positions of the non-zero bits of TS (or TC) to shift the internal terms of the sum of products in Q (or P).

The final architecture depends on all the parameters and the quantified coefficients of the P and Q polynomials provided by our Maple program. A complete example is presented in Section 4.

When using a degree-2 polynomial approximation, the square of h can be shared by the polynomials P and Q. This is the only "true" multiplication used in our method and it is not a full-width multiplication since h is only a l-bit value. This sharing may be used for higher values of d.

4. EXPERIMENTAL RESULTS

As the parameters space is very large, we focus in this section on a specific example, but our Maple program can generate the operator description for all parameters. In order to illustrate our results using plots, we use moderate precision: $w_I = w_O = 12$ bits. We fix the parameters to $m = 6$, $t = 4$ and $k = 2$. The total computation time of the Maple algorithms is limited to a couple of minutes for this example.

4.1. Numerical Results

The result of the quantification of the $\sin(a)$ table is illustrated on Figure 1. The quantification error for point a is measured by $\sin(a) - TS(a)$. The maximum error is around 8 LSBs (1 LSB is equal to 2^{-w_O}), which is limited w.r.t. the $t = 4$ bits constraint used to quantify the table values. The 8 LSBs maximal error means that the 3 last bits are wrong. Using $t = 3$ leads to a maximal error around 60 LSBs ($t = 2$ leads to 250+ LSBs maximal error). Similar results have been obtained for the $\cos(a)$ table.

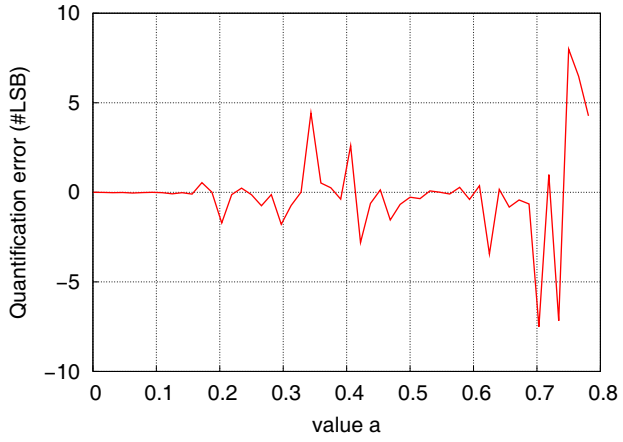

Fig. 1. Quantification error for $\sin(a)$ table.

The products of the quantified values in the TS and TC tables by the polynomials P and Q are performed using several shifts and additions/subtractions. Table 1 presents the distribution of all the possible shifts.

The quantification of the minimax polynomial approximations to $\sin(h)$ and $\cos(h)$ respectively gives:

$$P = x - \frac{x^2}{2^8} \quad \text{and} \quad Q = 1 - \frac{x}{2}.$$

s	TS	TC	s	TS	TC	s	TS	TC
1	24	11	5	17	15	9	13	17
2	14	11	6	21	15	10	13	9
3	19	10	7	13	18	11	6	8
4	17	12	8	11	11	12	7	1

Table 1. Number of s-bit shifts due to the quantified values stored in tables TS and TC.

The approximation error before and after quantification is reported in Table 2.

function	polynomial	minimax	after quantif.
$\sin(h)$	P	25.5	21.5
$\cos(h)$	Q	32.5	28.5

Table 2. Accuracy (number of correct bits) of the polynomial approximations to $\sin(h)$ and $\cos(h)$ before and after quantification of the coefficients ($0 \leq h \leq 2^{-6}$).

As the target accuracy is small, the quantified polynomial for $\cos(h)$ corresponds to the Taylor series expansion. But it is not usually the case (e.g., the approximation to the sine function in this example or example from Section 3.2).

The numerical quality of the final results is estimated by measuring the total error using for the sine function $\sin(x) - \sin_{impl}(a + h)$ where $\sin_{impl}(a + h)$ is the result of the actual implementation of equation (3). This total error of the proposed method for the sine function is illustrated on Figure 2. The total error corresponding to the cosine function is illustrated on Figure 3.

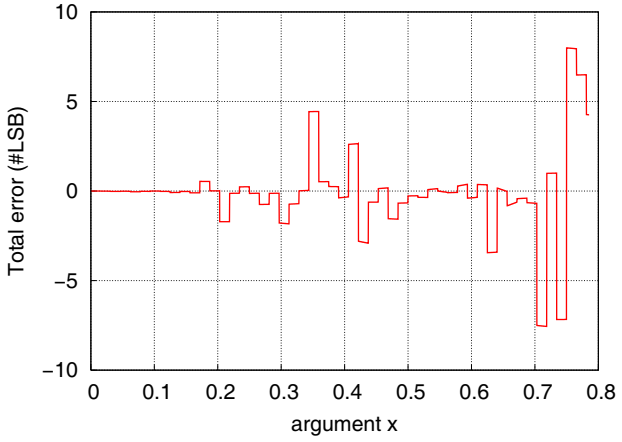

Fig. 2. Total error for $\sin(x)$ using the proposed method.

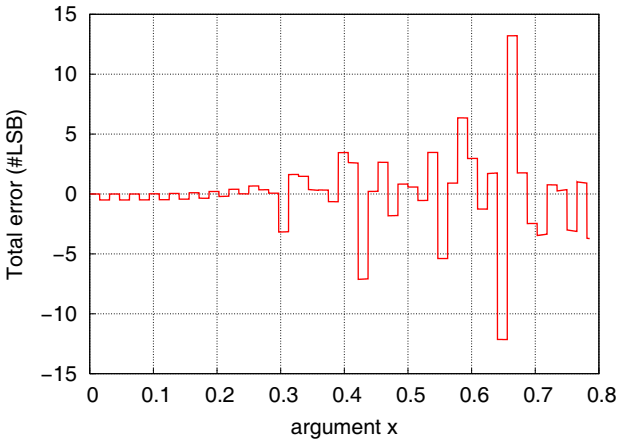

Fig. 3. Total error for $\cos(x)$ using the proposed method.

4.2. Implementation Results on FPGAs

We have implemented the operator on Virtex Xilinx FPGAs. The implementation results have been obtained using Xilinx ISE tools. Standard effort has been used both for synthesis and place-and-route (P&R) steps. The reported results are post-P&R values. Area and period are reported in number of slices and in nanoseconds respectively. The reported area includes all the tables implemented using the distributed memory ressources of the FPGA.

Table 3 presents the implementation results for the 12-bit simultaneous sine and cosine evaluation example. It also presents the comparison to two other methods for the approximation to the sine function: the multipartite method from [2] and the single multiplication second order (SMSO) from [4] for similar average accuracy. The reader should keep in mind that the two other methods only provide an approximation to the sine function and not the simultaneous sine and cosine functions. So, in practice, the proposed method shows both speed and area improvements even for a small accuracy.

method	area [# slices]	speed [ns]
this work (sin, cos)	79	16
multipartite (sin)	76	18
SMSO (sin)	59	17

Table 3. Implementation results and comparisons on Virtex FPGAs.

5. CONCLUSION

We have presented a method for moderate-precision approximation to the sine and cosine functions simultaneously in hardware. This method allows to evaluate both functions in about the same time as to compute only one of them. The obtained operators provide very small average error with reasonable maximum error. This makes our method suitable for some applications in digital signal or image processing.

The proposed method uses trigonometric identities, small lookup tables and low-degree polynomial approximations with very sparse coefficients. It leads to fast and small architectures. It requires at most one small multiplication, all other multiplications have been transformed into additions or subtractions. A Maple program generates all the polynomial coefficients and tables values used for a given set of parameters. The proposed method is efficient for simultaneous evaluation of sine and cosine functions, more efficient methods may be used when dealing with only one function (e.g. [4]).

In the future, we will study the relations between the parameters (w_I, w_O, m, d, k and t) and try to use centered interval for h.

6. REFERENCES

[1] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*, Birkhäuser, Boston, 1997.

[2] F. de Dinechin and A. Tisserand, "Multipartite tables methods," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 319–330, Mar. 2005.

[3] M. Schulte and J. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 842–847, Aug. 1999.

[4] J. Detrey and F. de Dinechin, "Second order function approximation using a single multiplication on FPGAs," in *14th International Conference on Field-Programmable Logic and Applications*. Aug. 2004, pp. 221–230, LNCS 3203.

[5] P. Markstein, "Accelerating sine and cosine evaluation with compiler assistance," in *Proc. of International Symposium on Computer Arithmetic (ARITH16)*, J.-C. Bajard and M. Schulte, Eds., Santiago de Compostela, Spain, June 2003, pp. 137–140, IEEE Computer Society.

[6] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, Morgan Kaufmann, 2003.

[7] E. Remes, "Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation," *C.R. Acad. Sci. Paris*, vol. 198, pp. 2063–2065, 1934.