Hardware Implementation of Math Module based on CORDIC Algorithm using FPGA

Muhammad Nasir Ibrahim
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia.

Mariani Idroas
Faculty of Petroleum and Renewable Energy Engineering
Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia.

Zuraimi Yahya
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia

Chen Kean Tack
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia

Siti Noormaya Bilmas
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia

***Abstract*— This paper discusses the implementation of math hardware module based on CORDIC algorithm to solve trigonometry, hyperbolic and exponential function on FPGA. CORDIC is one of the hardware efficient and iteration based algorithms that is used to implement various transcendental functions such as trigonometry, hyperbolic, exponential and so forth. In addition, by using this algorithm, the hardware requirement and cost are less as only shift registers, adders and ROM are required. Thus, the design is implemented on FPGA since it provides a versatile and inexpensive way for implementation. The design is then further interfaced with 4x4 matrix keypad and 16x2 character LCD to build a simple math hardware module for real time application. The coding of algorithm was written in Verilog HDL and the verification is done firstly by using simulation results of the ModelSim and then using the implementation on Altera DE1 board with the design interfaced with keypad and LCD to display the results.**

Keywords— ***FPGA, CORDIC, 4x4 matrix keypad, 16x2 character LCD, Verilog HDL.***

I. Introduction

With the state-of-the-art of the computer technology, the calculation operations of the processor must be always done in fast and precise way to avoid system crash or error. Therefore, it is important to implement a math module to solve transcendental functions of high precision with suitable hardware cost as well as to achieve high performance for the processor. Thus, the coordinate rotational digital computer (CORDIC) algorithm is used to achieve this target.

The Coordinate Rotational DIgital Computer (CORDIC) algorithm was first proposed by Jack E. Volder [2] in 1959 and then modified by J.S Walter [3] in 1971 to be a unified or more generalized algorithm. Thus, this algorithm is an iterative algorithm for the calculation of rotation of a two dimensional vector in linear, circular and hyperbolic coordinate systems. It is specially developed for real time digital computers where the computations mainly related to transcendental function such as trigonometry, hyperbolic [4][5], exponential [4] and logarithm. It provides advantages of low cost, less hardware requirements and simple for hardware implementation. Thus, the applications of this algorithm include digital signal and image processing especially for image rotation system [1]. Basically, this algorithm uses simple shift, add, subtract and look-up operations to perform computation in hardware.

In this paper, the design process of a simple CORDIC math hardware module and its interface with 4x4 matrix keypad and 16x2 character LCD was presented.

II. Methodology

To implement this module, CORDIC algorithm as well as its architecture must be fully understood. After that, the implementation was started by developing the design using Verilog code to model the CORDIC architecture. At this stage, the simulation was made using ModelSim to check the functionality of the modelled design. Then, the interface circuit was introduced into the design to display the correct results on LCD for further validation.

A. *CORDIC Algorithm and Architechture*

CORDIC algorithm uses simple shift, add, subtract and look-up operations to perform computation in hardware. Therefore, to design it, WE need some shift registers, adders, subtractors and ROMs. Generally, this algorithm is derived from the rotational transform equations as shown in equation (1) and (2).

$$X_n = X_0 \cos(\emptyset) - Y_0 \sin(\emptyset) \quad (1)$$

$$Y_n = Y_0 \cos(\emptyset) - X_0 \sin(\emptyset) \quad (2)$$

Thus, the simplified equations as shown below:

$$X_n = \cos(\emptyset)[X_0 - Y_0 \tan(\emptyset)] \quad (3)$$

$$Y_n = \cos(\emptyset)[Y_0 - X_0 \sin(\emptyset)] \quad (4)$$

By assuming that $\tan(\emptyset) = \pm 2^{-i}$ where i is the number of iteration, then the multiplication in the equation (3) and (4) replaced with simple shift operation. Therefore, the iteration equation becomes as shown in equation (5) and (6).

$$X_{i+1} = K_i[X_i - Y_i d_i 2^{-i}] \quad (5)$$

$$Y_{i+1} = K_i[Y_i - X_i d_i 2^{-i}] \quad (6)$$

$$where\ K_i = \cos(tan^{-1}(2^{-i})),\ d_i = \pm 1$$

After that, if the scaling factor, K_i is removed, the resulted equation will only consist of simple shift and add operation only. Thus, the value of K_i approaches 0.607252935 as the number of iteration approaches infinity. Therefore, the finalize iteration equation for CORDIC algorithm is shown in equation (7), (8) and (9).

$$X_{i+1} = X_i - Y_i d_i 2^{-i} \quad (7)$$

$$Y_{i+1} = Y_i - X_i d_i 2^{-i} \quad (8)$$

$$Z_{i+1} = Z_i - d_i tan^{-1}(2^{-i}) \quad (9)$$

$$where\ d_i = \begin{cases} -1, & Z_i < 0 \\ +1, & otherwise \end{cases}$$

Since the equation above can only solve for trigonometric function, J.S Walter [3] modified the original CORDIC equation into a unified CORDIC algorithm. It generalized several transcendental functions into a single algorithm. Thus, this algorithm defines a set of iteration equations to solve for trigonometry, hyperbolic and exponential functions by using the same hardware resources. The modified iteration equations are shown in the equation (10), (11) and (12).

$$X_{i+1} = X_i - m d_i 2^{-i} Y_i \quad (10)$$

$$Y_{i+1} = Y_i + d_i 2^{-i} X_i \quad (11)$$

$$Z_{i+1} = Z_i - d_i e(i) \quad (12)$$

where m is the decision factor for the coordinate system as shown in the following table.

TABLE I. COORDINATE SYSTEM OF UNIFIED CORDIC AND ITS CORRESPONDING E(I) FUNCTION [3]

m	Coordinate system	Value of e(i)
1	Circular	$tan^{-1}(2^{-i})$
0	Linear	$tanh^{-1}(2^{-i})$
-1	Hyperbolic	2^{-i}

TABLE II. UNIFIED CORDIC IN ROTATIONAL MODE [3]

m	Rotational Mode d_i = sign(Z_i), Z_i rotate towards 0
1	For cos and sin, set X_0 = 1/K, Y_0 = 0 where K = 1.646760258121.. $X_n = \cos(Z_i), Y_n = \sin(Z_i)$ $\tan(Z_i) = Y_n/X_n$
0	For multiplication, set Y_0 = 0 $X_n = X_0, Y_n = Y_0 + X_0 Z_0 = X_0 Z_0$
-1	For cosh and sinh, set X_0 = 1/K', Y_0 = 0 where K' = 0.8281339907.. $X_n = \cosh(Z_i), Y_n = \sinh(Z_i)$ $\tanh(Z_i) = Y_n/X_n$ $e^x = X_n + Y_n$

Then, by introducing initial values for X and Y and providing the input to Z, the CORDIC operation is started and compute the values of X, Y and Z for the next iteration based on the equations (10), (11) and (12) and continue until specific number of iteration reaches.

Therefore, the basic architecture of the CORDIC algorithm is shown in Figure 1.

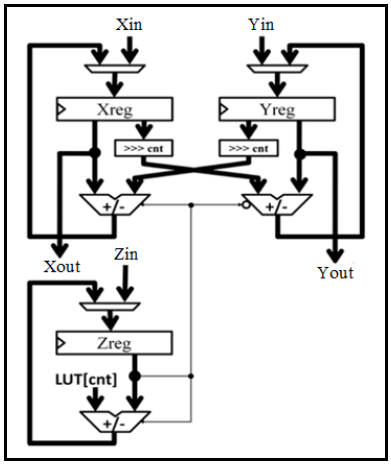

Fig. 1. The basic architecture of CORDIC algorithm for one iteration

B. CORDIC Architecture Modelling by Verilog HDL

By using Quartus II software with Verilog HDL coding style, the CORDIC architecture as shown in Figure 1 can be modelled and implemented. In this paper, we implemented CORDIC algorithm to solve trigonometry, hyperbolic and exponential. Thus, the basic steps to code the CORDIC for trigonometry and hyperbolic functions are shown below:

(1) Set the value of shifted X (X_shr) to a value after shifting X right by i places.
(2) Set the value of shifted Y (Y_shr) to a value after shifting Y right by i places.
(3) Set the value of delta Z (Z_shr) from the values in LUT and set the value of m according to the equations as shown in Table 1 (m = 1 for trigonometry and m = -1 for hyperbolic).
(4) Determine the rotation direction and the values of X, Y and Z for next iteration. Two cases to determine:
   (i) If Z >= 0, rotate the angle in anti-clockwise direction for the next iteration. Thus, set X to value of X – m*dY, set Y to value of Y + dX and set Z to value of Z – dZ in order to update the values for X, Y and Z.
   (ii) If Z < 0, rotate the angle in clockwise direction for the next iteration. Thus, set X to value of X + m*dY, set Y to value of Y – dX, set Z to value of Z + dZ in order to update the values for X, Y, and Z.

Then, to evaluate the value of exponential function [4], we use an adder to connect the output of X and Y from hyperbolic mode (m = -1) after n iteration to perform addition between these values since e^x = cosh x + sinh x. The block diagram is shown in Figure 2.

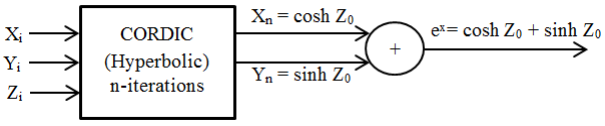

Fig. 2. Block diagram of the exponential function determination

To improve mathematical throughput or increase the execution rate, calculations for fractional values can be performed by using unsigned fixed-point representations or two's complement signed fixed-point representations [8]. Thus, it requires the programmer to create a virtual decimal place for a given length of data. For this purposes, Q format can be used to realize it. The convention is as shown in the following:

$$Q\ [m].[n] \tag{13}$$

where

m = number of integer bits (including the sign bit for signed number)
n = number of fractional bits
m+n = Total bits of the representation
= number of integer bits + number of fractional bits

However, to have higher precision for the output, the IEEE-754 single precision 32-bits floating point format was used to represent the floating point number which converted from Q-format. According to IEEE-754 standard, the data for this format has 1 bit of sign bit (S), 8 bits of biased exponent (E) and 23 bits of mantissa (M) as shown in Figure 3.

Fig. 3. IEEE-754 Single Precision Formats [7]

Thus, this format represented a floating point number based on following equations:

$$Value = \begin{cases} (-1)^S \times 2^{(E-Bias)} \times 1.M\ (normalized), & E > 0 \\ (-1)^S \times 2^{(E-Bias)} \times 0.M\ (denormalized), & E \leq 0 \end{cases}$$

where

$$M = m_{22} \cdot 2^{-1} + m_{21} \cdot 2^{-2} + m_{20} \cdot 2^{-3} + \cdots + m_1 \cdot 2^{-21} + m_0 \cdot 2^{-20}$$

S = Sign bit (1 or 0)
E = Biased exponent (0 to 255)
Bias = 127

C. Interface Circuit Designs

After the CORDIC algorithm is modelled, we introduced an external interface circuit which consist of 4x4 keypad and 16x2 character LCD to the design for real time application. Thus, we have soldered a simple circuit for these interface on a donut board and then connected to the GPIO ports of Altera DE1 board for interfacing. After that, to interface keypad and LCD with the Altera FPGA board, we need to design the controllers using Verilog HDL to control the interface operations of these peripherals.

1) 4x4 Matrix Keypad Interface

To interface the keypad with DE1 board, the rows and columns pins are connected to the GPIO pins of the DE1 board and make the proper pin assignment. Then, a keypad scanner was needed to scan which button is pressed. Thus, we need to scan it column by column and row by row every certain short period. The row pins should be connected to input port and then the column pins are connected to the output port. At the same time, the row pins need to pull up or pull down with resister to avoid floating case happen. Thus, the basic block diagram for 4x4 matrix keypad is shown in Figure 4. In addition, a de-bouncer was also required to filter out the glitches associated with switch transitions. Thus, we used a timer to generate a one-clock-cycle enable tick every 10ms and then introducing finite state machine (FSM) to keep track of whether the input is stabilized.

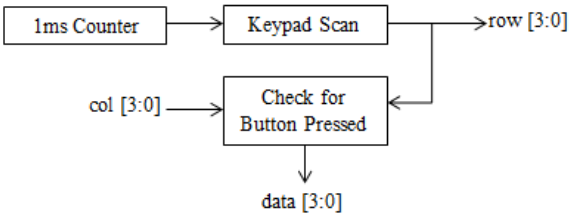

Fig. 4. Block diagram of Matrix Keypad

2) LCD Interface

To interface character LCD module with DE1 board, the LCD pins are connected to GPIO pins in the DE1 board and then make proper pin assignment. Then, the specific command data in 1 byte as shown in Table 3 was sent to the LCD to perform certain operations in command mode (RS = 0) such as clear display, set entry mode, set display address. Meanwhile, to write specific characters or symbols on the LCD, the operation is made in write mode (RS = 1). Then, the ASCll code for several characters and symbols were sent to the LCD one by one at each address of LCD.

TABLE III. COMMON COMMAND DATA

Command data (in binary)	Descriptions
00111000	Function Set for 8 bits data transfer and 2 line display
00001111	Display On, without cursor
00000001	Clear Screen
00000110	Entry mode set, increment cursor automatically after each character was displayed
00000010	Return the cursor to home address

3) Overall Design Architecture

Finally, we combined all the designs and make it as a system to generate the output results on LCD interface based on the selection of the user. Thus, the overall design architecture is shown in Figure 5.

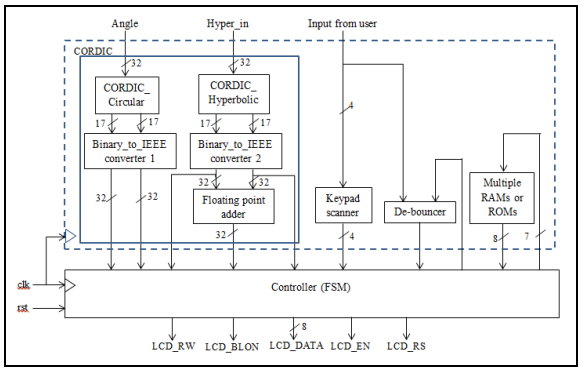

Fig. 5. Overall design architecture

III. Result and analysis

A. Simulation Result

The output waveform for CORDIC module is shown in Figure 6.

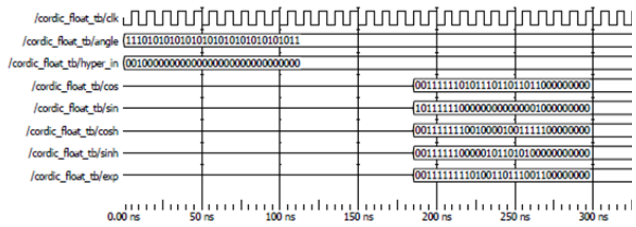

Fig. 6. Simulation result of CORDIC module

Figure 6 shows the output waveform generated by CORDIC module and its values were tabulated in Table 4. It performs the CORDIC iteration calculations and gives the results of cos, sin, cosh, sinh and exp. The input data are represented in Q-format and the output data are represented in IEEE-754 single precision floating point format. Meanwhile, this design requires about 18 clock cycles or latency for computation as shown in Figure 6 due to iteration calculations in the CORDIC algorithm.

TABLE IV. THE RESULTS WITH ITS FORMAT TYPE AND DECIMAL VALUES

Signal Name	Output (in hex form)	Format	Decimal values
angle	EAAAAAAB	Q0.32 unsigned	330 degree
hyper_in	20000000	Q2.30 unsigned	0.5
cos	3F5DB600	IEEE-754 32-bits	0.86605835
sin	BF000200	IEEE-754 32-bits	-0.5000305
cosh	BF909F00	IEEE-754 32-bits	1.1298523
sinh	3F05A800	IEEE-754 32-bits	0.5220947
exp	3FD37300	IEEE-754 32-bits	1.651947

Therefore, if compare the results with the actual answers calculated by scientific calculator, the results of my design show the precision of approximately 2-4 decimal places and above. Thus, it shows that the results are correct but it might

need further improvement in precision especially for hyperbolic part.

B. LCD Results from Interface Circuit

Based on the CORDIC module, we further interface with an I/O interface circuit to display the result so that we can check the results more easily without tracing from the simulation waveform. Thus, Figure 7 shows the completed I/O interface circuit that has been done on the donut board.

Fig. 7. I/O interface circuit on donut board with working LCD display

By using this module, we display the results in hexadecimal form that converted from the binary value of 32-bits single precision IEEE-754 floating point format. Thus, by introducing some inputs from the CORDIC module as discussed in previous section where angle = 330^{o} and hyper_in = 0.5, the outputs on LCD displays for cos, sin, cosh, sinh and exp were recorded as shown in Table 5.

TABLE V. RESULTS COLLECTED FROM LCD DISPLAY OUTPUTS

Output Functions	Outputs on LCD displays (in hex form)
cos	0x3F5DB600
sin	0xBF000200
cosh	0x3F909F00
sinh	0x3F05A800
exp	0x3FD37300

Based on the results on Table 5, since the values that displayed on the LCD were totally the same with the simulation values, it means that the interface circuit is working and the results were verified.

IV. Conclusions

Based on the results obtained from the LCD display of the interface circuit, it shows the same results as obtained from the simulated results but the number was converted to hexadecimal form due to insufficient spaces on LCD to display the 32 bits binary number in single line. Thus, the numbers that are displayed on LCD were shorter and easier to read. So, we can use this circuit to test the functionality of the design without referring to the simulation waveform.

In a nutshell, we successfully design a math hardware modules based on CORDIC algorithm to solve trigonometric, hyperbolic and exponential with an acceptable precision and implement on Altera DE1 board with the simple working I/O interface circuit.

References

[1] D. Yi-Jun, B. Zhuo, "CORDIC algorithm based on FPGA," Journal of Shanghai University, vol. 15, no. 4, pp 304-409, Aug 2011.

[2] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electronic Computers, vol. EC-8, no. 3, pp. 330-334, Sept. 1959.

[3] J. S. Walther, "A unified algorithm for elementary functions," in AFIPS Spring Joint Computer Conference, vol. 38, pp 379-85, 1971.

[4] Boudabous, A., Ghozzi, F., Kharrat, M.W., Masmoudi, N., "Implementation of hyperbolic functions using CORDIC algorithm," The 16th International Conference on, pp.738, 741, 6-8 Dec. 2004.

[5] Shrugal V., Nisha S., Richa U., "Hardware Implementation Of Hyperbolic Tan Using Cordic On FPGA," International Journal of Engineering Research and Application, vol. 3, no. 2, pp. 696-699, March – April 2013.

[6] Tanya V., David E., Sven K., Martin S., "Floating-point Mathematical Co-Processor for a Single-Chip On-board Computer," Surrey Space Centre, School of Electronics and Physical Sciences, University of Surrey, Guildford, UK.

[7] Meenu T., Karan G., Sharmelee T., "Performance analysis of Floating point adder using Sequential Processing on Reconfigurable hardware," International Journal of Engineering Research and Application, vol. 2, no. 3, pp. 1226-1229, May-Jun 2012.

[8] Chun T. E., Peter Y.K., George A, "Dual Fixed-Point: An Efficient Alternative to Floating-Point Computation," International Conference and Field Programmable Logic, pp. 200-208, 2004.