CORDIC Accelerator for RISC-V

Recep Onur Yıldız
Computer Engineering Department
Istanbul Technical University
Istanbul, Turkey
yildizr@itu.edu.tr

Ayse Yilmazer-Metin
Computer Engineering Department
Istanbul Technical University
Istanbul, Turkey
yilmazerayse@itu.edu.tr

2021 29th Telecommunications Forum (TELFOR) | 978-1-6654-2585-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/TELFOR52709.2021.9653439

***Abstract*—Software Defined Radio (SDR) is a highly configurable transceiver that provides flexibility in communication systems. The flexibility of SDR is achieved by implementing most of the SDR architecture in software. The software of the SDR is responsible for the modulation/demodulation of messages and frequency conversion of the signals. These operations frequently use Sine and Cosine functions. Sine and Cosine functions are required to be stored in memory or to be calculated. Storing these values in the memory causes a decrease in core performance. Sine and Cosine functions can be calculated with Coordinate Rotation Digital Computer (CORDIC) algorithm or Taylor's Series. Taylor's series consists of multiplication and division operations which are hard to implement in hardware. This study proposes an accelerator that calculates Sine and Cosine functions with CORDIC algorithm. The architecture of the accelerator is based on systolic array architecture to allow the core to acquire the functions' value immediately. The result shows that the proposed system provides at least 32x speed up over Taylor's Series having grade 4. While providing speedup, the proposed accelerator increases the resource usage of the bare system as around 5,5% and power consumption of bare system as around 8%.**

***Index Terms*—sdr, risc-v, cordic. accelerator**

I. Introduction

Advance in technology leads to growth in the number of active wireless devices. These devices communicate with each other using communication protocols. Each of these communication protocols has unique characteristics such as efficiency and frequency. The communication protocol of the device is implemented by transmitter and receiver systems. When any modification on the communication protocol is necessary, these systems may require to be modified also. To realize the modifications on these systems, the hardware needs to be modified if transmitter and receiver are implemented in hardware. But, modifying hardware takes time and needs money. To overcome this problem Mitola [1] proposed the Software Defined Radio (SDR) concept. The proposed architecture implements transmitter and receiver systems as a collection of hardware and software. There are 3 segments in SDR which are RF Front End, Digital Front End and Signal Processing. The RF Front End which is responsible for transmitting and receiving communication signals is implemented in hardware. Digital Front End and Signal Processing segments are implemented in software. Digital Front End is responsible for conversion of baseband signal to Intermediate Frequency (IF) signal or vice versa. The modulation and demodulation of messages are performed in Signal Processing segment. Thus, any change in the implemented communication protocol can be done in the software of SDR.

Software of SDR can be implemented in Field Programmable Gate Array (FPGA) or General Purpose Processor (GPP). FPGA is a highly re-configurable architecture and it allows users to design digital systems. GPP is a processor that is re-programmable and allows to run software sequentially. Designing an SDR system on GPP provides ease of development. The software of GPP is translated into instructions that are defined by the Instruction Set Architecture (ISA). RISC-V is a free and open ISA that allows efficient implementations by avoiding over-architecting [2]. RISC-V ISA defines a base integer ISA and it can be extended in two ways; standard extension and non-standard extension. The non-standard extensions use reserved opcodes for custom instructions that can be used to control an accelerator. Attaching an accelerator to the RISC-V core, increase the low computing performance of the GPP originated from the sequential processing.

Trigonometric functions can be stored in memory but storing these functions has an important area penalty. In addition to storing, Taylor Series can be used to calculate these functions [3]. The accuracy of the function increases when the grade of Taylor Series increases. The sine function can be calculated with Taylor Series as in the Eq. 1. The calculation includes multiple divisions, multiplications, subtractions, and additions which may cause pipeline stalls in the core.

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots \tag{1}$$

To acquire these functions, Jack E. [4] proposed a technique called Coordinate Rotation Digital Computer (CORDIC). CORDIC is suitable for real-time applications which mainly consist of trigonometric functions. Since the CORDIC algorithm uses shifting and addition operations in calculations, it is cost-efficient to implement in hardware.

In this study, a novel CORDIC accelerator for RISC-V ISA is proposed. The CORDIC algorithm is implemented in a systolic array architecture. Systolic array architecture decreases the calculation time with parallel operations and finishes the calculations of CORDIC in a single cycle. The attached CORDIC accelerator increases the computation throughput of GPP and provides flexibility to RISC-V core in Sine and Cosine calculation. This flexibility makes the RISC-V core

978-1-6654-2585-8/21/$31.00©2021 IEEE

more convenient to use in SDR architecture. This document is organized as follows: Section II provides information on related work. The architecture of the proposed system is described in Section III. Section IV presents the evaluation results of the design. This paper is concluded in Section V.

II. Related Work

RISC-V ISA provides features to study and conduct research on computer architecture. The base integer ISA allows standard and non-standard instruction extensions. An accelerator can be attached to the core and ISA can be extended with non-standard instructions to control the accelerator. In [5], the authors presented a configurable DSP accelerator that accelerates addition, multiplication, and linear combination operations. The authors of [6] presented a parameterized Secure Hashing Algorithm (SHA) accelerator for RISC-V. SHA algorithm is a widely used hashing algorithm that takes an input message and generates the unique hash value for the related input. The parameterization of the SHA3 accelerator helps the designer to evaluate the parameterized accelerator in energy efficiency, performance, and size. In [7], the authors presented a configurable system generator called Gemmini. Gemmini generator supports a variety of different hardware architecture, programming interface, system integration options to generate Deep Neural Networks (DNN) accelerator. The central unit of Gemmini is based on systolic-array architecture and it is responsible for the calculation of the accelerator.

CORDIC algorithm is a resource-efficient implementation of trigonometric function calculation. Trigonometric functions are calculated by addition and shifting operations which are cheap for hardware implementation. In [8], the authors presented a re-configurable CORDIC concept. The proposed concept can be configured to operate in either circular or hyperbolic trajectories. The mode of the proposed CORDIC is also configurable as rotation or vectoring-modes. The authors of [9] presented an efficient CORDIC algorithm by removing redundant iterations of CORDIC algorithm. In the proposed design, conventional CORDIC and Virtually Scaling-Free (VSF) CORDIC is combined. VSF CORDIC follows the same direction in vector rotation.

III. System Architecture

In this study, CORDIC accelerator for RISC-V is proposed. The accelerator is tightly coupled to the core and it communicates with the core through a custom interface which is Rocket Custom Coprocessor (RoCC) interface. The system architecture is illustrated in Fig. 1.

A. Architecture of Rocket Core

The Rocket Core is generated with the Chipyard framework which is used to generate System on Chip (SoC) systems. Chipyard framework includes Rocket Chip which is a SoC generator developed in Berkeley. Rocket Chip generates the Rocket Core and SoC parts besides the Rocket Core. The Rocket Core is a core that is a 5-stage in-order RISC-V core. The Rocket Core implements the RV32GC and RV64GC

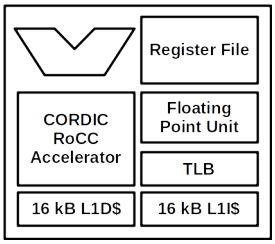

Fig. 1. The system architecture of Rocket Core with CORDIC accelerator

31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
0000000000000000	src	SINE	dest	custom-2
0000000000000000	src	COSINE	dest	custom-2

Fig. 2. Instruction format

ISAs [10]. The abbreviation ”G” represents general-purpose ISA which combines integer multiplication, atomic instruction, single precision, and double precision floating point extensions with base integer ISA. The abbreviation ”C” represents the 16-bit instruction format. In this study, RV64GC ISA is selected as the ISA of the core. The ISA of the proposed system is extended as the reserved non-standard custom instruction of RISC-V ISA encoding space. The format of the proposed custom instruction is in the Fig. 2 When the function of instruction is equal to 0 (1), the accelerator returns the cosine (sine) function result.

In the proposed design, input and output format of the accelerator is fixed-point binary representation which consists of integer part and fraction part. The most significant 32 bits of 64 bits represent the integer part of fixed-point. The least significant 32 bits of 64 bits are the fraction. The core sends the angle input in degree and the accelerator sends the output of the selected function over the RoCC interface. The RoCC interface includes the response of the accelerator and command by the Rocket Core. The Rocket Core sends opcode, function, source registers, and destination register and the accelerator responds with data and destination register. In addition to that signals, there are ready and valid signals between the accelerator and Rocket Core. When the ready signal is active, the corresponding RoCC signals are shared and the valid signal is activated. The calculation of the CORDIC is finished in one clock cycle and the output is shared via RoCC interface in one cycle. So that, the core can have the sine or cosine function result in two clock cycles by using the proposed CORDIC accelerator.

B. Architecture of CORDIC Accelerator

CORDIC is a technique that calculates trigonometric functions. CORDIC algorithm rotates the given coordinates step by step until the angular argument of the vector is zero. The accuracy increases when the number of rotation increase. Following equations hold when the given vector is a unit vector and its coordinates are as in Eq. 2 and Eq. 3.

$$Y = sin\theta \tag{2}$$

$$X = cos\theta \tag{3}$$

In the $(i+1)$th step of CORDIC algorithm, coordinates of ith vector are rotated as α. The coordinates of $(i+1)$th vector are calculated as in Eq. 4 and Eq. 5.

$$X_{i+1} = X_i cos\alpha - Y_i cos\alpha = cos\alpha(X_i - Y_i tan\alpha) \tag{4}$$

$$Y_{i+1} = Y_i cos\alpha + X_i sin\alpha = cos\alpha(Y_i + X_i tan\alpha) \tag{5}$$

The iterative rounds add a multiplier as $cos\alpha_i$ where α_i represents the rotation angle. The rotation angle in each round is defined as in Eq. 6.

$$\alpha_i = tan^{-1}(2^{-i}) \tag{6}$$

Since the $cos\alpha_i$ and $cos(-\alpha_i)$ are equal and the rotation angles are predefined, the cosine multipliers can be combined as in Eq. 7. The n represent the total round number in CORDIC algorithm. For the 8 round CORDIC algorithm K is calculated as 0.607261323.

$$K = \prod_{i=0}^{n-1} ki = cos\alpha_0 cos\alpha_1 cos\alpha_2 \cdots cos\alpha_{n-1} \tag{7}$$

In the $(i+1)$th round, the coordinates are calculated in Eq. 8 and Eq. 9 since $\alpha_i = tan^{-1}(2^{-i})$. The operation is defined by the d_i whether it is an addition or a subtraction. The d_i can be equal to +1 or -1 which is defined by the sign of the z_i. When the sign of d_i is equal to the sign of z_i. Power of 2's in the equations can be realized by right shift in hardware because shifting right is equal to 2^{-1}. So that, computations of CORDIC are implemented by shifting and addition/subtraction operations. In the first step of the CORDIC algorithm, $i = 0$, $\alpha = 45$, $X_0 = K$, $Y_0 = 0$, and , $Z_0 = \theta$ where θ is the input angle.

$$X_{i+1} = X_i - d_i 2^{-i} Y_i \tag{8}$$

$$Y_{i+1} = Y_i + d_i 2^{-i} X_i \tag{9}$$

$$Z_{i+1} = Z_i - d_i \alpha_i \tag{10}$$

The CORDIC architecture can be implemented as word-serial or pipelined [11]. The word-serial implementation rotates the angle in each cycle and rotations are unfolded in pipeline implementation. In this study, a novel CORDIC accelerator is implemented in systolic array architecture. The proposed system calculates the sine or cosine functions in a single clock cycle. Therefore, possible stalls of core during these calculation is prevented. The systolic array architecture contains interconnected Processing Elements (PE). Each PE is responsible for the dedicated simple calculation and operates parallel with other PEs. Computing simple calculations in PE makes the main calculation less complex [12]. All PEs in the CORDIC accelerator are identical which have 4 inputs named as X, Y, C, and S; and 2 outputs named as Result and Carry. The X and Y inputs of PE are the data inputs and C is the carry input. The input S determines the type of operation whether it is addition or subtraction. The computation of PE are in the Eq. 11 and Eq. 12 where & represents logical-AND and $\wedge$ represents logical-XOR. When the input S is logical high (low), the PE subtracts (adds) input Y from input X. The representation of PE is in the Fig. 3. The interconnected PEs are responsible for the addition and subtraction in CORDIC algorithm. Shifting operation of CORDIC is implemented as the interconnection between PEs.

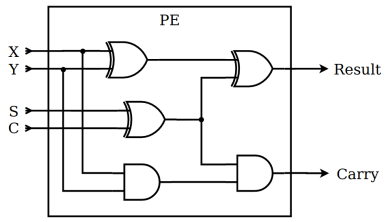

Fig. 3. PE used in CORDIC Systolic Array

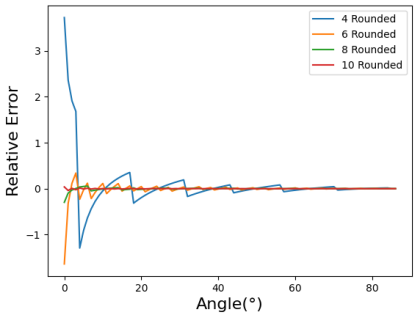

Fig. 4. Error rates of 4,6,8,10 rounded CORDIC

$$Result = X \wedge (Y \wedge S) \wedge C \tag{11}$$

$$Carry = X\&(Y \wedge S)\&C \tag{12}$$

CORDIC accelerator takes *angle* and *function* as its inputs. *Function* defines whether the trigonometric function is sine or cosine. CORDIC accelerator generates the selected trigonometric function output in a single clock cycle. The most significant 29 bits of integer part of the angle input and integer part of trigonometric function result are always logic-0. Therefore, the related PEs of these redundant bits are removed to decrease resource usage and power consumption.

IV. Test Results

In order to evaluate the effect of the number of rounds in CORDIC, the different numbers of rounded CORDICs are tested. The angle inputs of CORDICs are incremented as 1°from 0°to 90°. To visualize the round effect, the Eq. 13 is calculated for each round and plotted in the Fig. 4. When the round number of CORDIC increases, the output becomes more precise but resource usage increases. In this study, the round number is selected as 8.

$$\%Error = \frac{(RealSine - CORDICSine)}{RealSine} \tag{13}$$

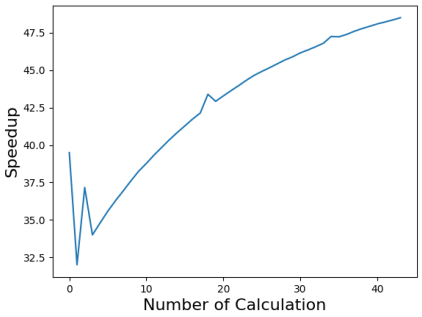

Fig. 5. Speedup of proposed system over 4 termed Taylor Series

TABLE I: Comparison of resource usage

Resource	Bare System	Propsed System
Slice LUTs	42858	45216
Slice Registers	20326	20431

Comparing the systolic array based proposed CORDIC implementation with 8 rounded parallel and iterative implementations of CORDIC, the proposed CORDIC implementation finishes the calculation in a single clock cycle. Nevertheless, conventional 8 rounded iterative CORDIC implementations finish the calculation in 8 clock cycles and 8 rounded parallel CORDIC implementations calculate the first value in 8 clock cycles [13]. On the other hand, parallel and iterative methods have lower path delay comparing to the presented architecture. Therefore, the proposed system has lower clock frequency. Moreover, the calculation time of 4 termed Taylor Series calculates the Sine function is tested. The calculation clock cycle of Taylor Series and proposed CORDIC accelerator is compared and the speedup of the proposed accelerator is illustrated in Fig. 5. The speedup of the accelerator increases when the number of consequent Sine calculations increases. The proposed system provides at least 32x speed up over a system that use 4 termed Taylor Series to calculate sine function. Since there are multiple additions, multiplications, and divisions in the Taylor Series computation, the calculation of Sine occupies the related ALUs during computation. Also, cache lines are occupied by the intermediate results of the calculation.

The proposed system and default system are compiled with Chipyard framework. The generated outputs of both systems are implemented using Vivado. The FPGA of the test system is Xilinx XC7A200. The implemented default system does not include stored Sine values and the proposed system implements the 8 rounded CORDIC aglorithm. The comparison of the resource usage is in the Table I. The implementation reports of Vivado show that adding the proposed CORDIC accelerator in the system increases the power consumption by 8%.

V. Conclusion and Future Works

In this study, a novel CORDIC accelerator is designed as in systolic array architecture. The accelerator is attached to the core and it communicates with the core via RoCC interface. In order to calculate the sine and cosine functions, RISC-V base integer ISA is extended. Using the added custom instruction, the core shares the input angle and function with the proposed accelerator. The function defines the trigonometric function whether it is Sine or Cosine and input angle is in degree. The CORDIC accelerator sends the calculated trigonometric function value to the core.

We evaluate the accuracy of the accelerator by changing the number of rounds in CORDIC. Moreover, the performance of the accelerator is tested and compared with the 4-grade Taylor Series implementation. The resource usage of the accelerator implementation is also compared with the default system.

In the future, we plan to add input/output (I/O) capability to accelerator. The accelerator will be able to drive a Digital to Analog Converter (DAC). It is also planned to extend the accelerator operands with built-in modulators and demodulators. Therefore, the system will be more convenient to be used in an SDR architecture.

References

[1] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, vol. 33, pp. 26–38, 1995.

[2] K. A. Andrew Waterman, "The risc-v instruction set manual," *Annalen der Physik*, vol. 322, no. 10, pp. 891–921, 1905.

[3] C. Brunelli, H. Berg, and D. Guevorkian, "Approximating sine functions using variable-precision taylor polynomials," in *2009 IEEE Workshop on Signal Processing Systems*, pp. 057–062, 2009.

[4] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959.

[5] L. Calicchia, V. Ciotoli, G. C. Cardarilli, L. di Nunzio, R. Fazzolari, A. Nannarelli, and M. Re, "Digital signal processing accelerator for risc-v," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 703–706, 2019.

[6] C. Schmidt and A. Izraelevitz, "A fast parameterized sha3 accelerator," tech. rep., EECS Department, University of California, Berkeley, 2015.

[7] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," 2021.

[8] S. Aggarwal, P. K. Meher, and K. Khare, "Concept, design, and implementation of reconfigurable cordic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1588–1592, 2016.

[9] Y. Xue and Z. Ma, "Design and implementation of an efficient modified cordic algorithm," in *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, pp. 480–484, 2019.

[10] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," Tech. Rep. UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

[11] M. D. Ercegovac and T. Lang, "Chapter 11 - cordic algorithm and implementations," in *Digital Arithmetic* (M. D. Ercegovac and T. Lang, eds.), The Morgan Kaufmann Series in Computer Architecture and Design, pp. 608–648, San Francisco: Morgan Kaufmann, 2004.

[12] Kung, "Why systolic architectures?," *Computer*, vol. 15, no. 1, pp. 37–46, 1982.

[13] M. Chinnathambi, N. Bharanidharan, and S. Rajaram, "Fpga implementation of fast and area efficient cordic algorithm," in *2014 International Conference on Communication and Network Technologies*, pp. 228–232, 2014.