A Comparison of Piplined and Parallel CORDIC Architecures

Mohamed.M.Elgazzar, Mahmoud Maher El-Sayed Mohammed

Abstract: ***Coordinate Rotation Digital Computer (CORDIC) is efficient and simple algorithm that is used to calculate the trigonometric and hyperbolic functions. The CORDIC is used in multiple applications like hardware multiplier in the field programmable gate array (FPGA) or in simple microcontroller. The advantage of the CORDIC is that it uses only shift and add operations. The CORDIC can be implemented in different architectures like the parallel (or combinational) architecture, pipelined architecture, and iterative architecture. We want to know what is the pros and cons for every architecture, so our research will be very important to the integrated circuit (IC) designer. The designer should know the advantages of each architecture to choose the suitable architecture for his design. In this paper we will make comparison between pipelined and parallel CORDIC architectures. The comparison steps are implementing the parallel and the pipelined CORDIC architectures in Verilog register transfer level (RTL) codes, then simulating these two Verilog codes using Modelsim. The both architectures will be simulated using the same test sequence. The sequence flow is inserting angles from zero degree to ninety degree and compare the outputs which are cosine and sine for the inserted angle to the expected values. The test sequence will be the same for both architectures. This will make the same simulation environment for the two architectures. From simulation results, we can expect the parallel CORDIC is more efficient in the initial delay, area, power consumption, and the pipelined CORDIC is more efficient in the clock frequency, critical path.***

Keywords: CORDIC, E. Volder, FPGA, Parallel CORDIC, Pipelined CORDIC.

I. INTRODUCTION

The first research in CORDIC algorithm was in aero electronics departmentby Jack E. Volder. This research aims to replace the analog resolver to digital one in the B-58 Bomber.Volder's research propose CORDIC algorithm that is used to solve the trigonometric calculations (sine, cosine). The research also includes the possibility to make coordinate rotation [1].

CORDIC algorithm depends on shift, and add operations for several Calculation tasks like hyperbolic, trigonometric, eigenvalue estimation, and square root calculation. There are several applications that we used CORDIC inside its hardware like image processing, communication devices, and 3D graphics [2].

The CORDIC algorithm is faster than the other techniques if the multiplier is not existing in the target hardware. The CORDIC will take area less than the area of the multiplier. Area is very importantin the design of IC.Also, the CORDIC will take time less than the time of software running on this processor to make these calculation tasks.

Revised Manuscript Received on 28 December 2018.

Mohamed. M. Elgazzar, Associate Professor, Higher institute of Computer Science and Information Systems, New Cairo, Egypt.

Mahmoud Maher El-Sayed Mohammed, Engineering Collage, Cairo University, Giza, Egypt.

If the processor did not contain multiplier inside it, then the CORDIC will help if we need to make parallel mathematical calculations. This will help the processor and make it free for other commands [8].

In the Internet of Things (IoT), the CORDIC is added with the processor that is used in the FPGA design implementation or in silicon IC. The limitation in the power and area push the companies to depend on weak processors. These processors have an advantage of low power consumption and occupy small area, but it has disadvantage that it is not powerful in the mathematical calculations. These processors will need CORDIC help to execute the hyperbolic, trigonometric mathematical operations [7].

II. CORDIC DESIGN OPERATIONS

A. Rotation Mode

In this mode the CORDIC is used to calculate the sine and cosine trigonometric functions simultaneously, polar to Cartesian coordinates conversion. We can calculate the trigonometric function by rotating the input vectors. In the rotation mode, The CORDIC starts with initializing the angle accumulator with input rotation angle. The rotation direction in every iteration is chosen to decrease the magnitude of residual angle in the angle accumulator. The CORDIC decide the rotation direction dependent on the sign of the residual angle [6].

B. Vectoring Mode

This mode is used if we want to calculate the angle of the input vector. The angle is calculated by make rotations on the input vector. In this mode CORDIC will rotate the vector using previously defined angle in the CORDIC algorithm. The aim of these rotation is to reduce the value of y coordinate till it become near to zero. In every iteration CORDIC decide the rotation direction according to the sign of residual y coordinate that we calculate in the previous iteration [4].

III. CORDIC ARCHITECTURES

In the hardware we can implement the algorithm in different architectures. For the CORDIC, we have three different architectures. The first architecture is the parallel (or combinational) which is pure combinational architecture. When the input is changed, the output will be ready in the next clock edge. The second architecture is the pipelined architecture which we use register after each stage to divide the critical path into small paths. The output of pipelined architecture will be ready after changing the input by number of clock

cycles equal the number of stages. The third architecture is the iterative architecture which we use one calculation stage of the pipelined architecture and make state machine which control the next input to the calculation stage [3] [5].

IV. PIPLINED AND PARALLEL CORDIC ARCHITECURE COMPARISON

In this Comparison we use Modelsim to make our simulations. We implement the pipelined and parallel CORDIC using Verilog. In the pipelined CORDIC we put pipeline stages in the RTL. These pipeline stages make the pipelined CORDIC Verilog code bigger than the parallel CORDIC Verilog code. We used same test benches for both CORDIC architectures but with different checkers due to the checker for the pipelined will be delayed by 14 clock cycles. Then we compared the output to the expected values to make sure that the both parallel and pipelined CORDIC architectures work right. Here is comparison from our simulation results:

A. Output Delay Comparison

The output delay of the parallel CORDIC in our simulationis shown in figure 1. The delay from the changing input to the valid output is one clock cycle. The output delay of the pipelined CORDIC in our simulationis 14 clock cycles as shown in figure 2. This can show that the parallel CORDIC has less initial delay than the pipelined CORDIC.

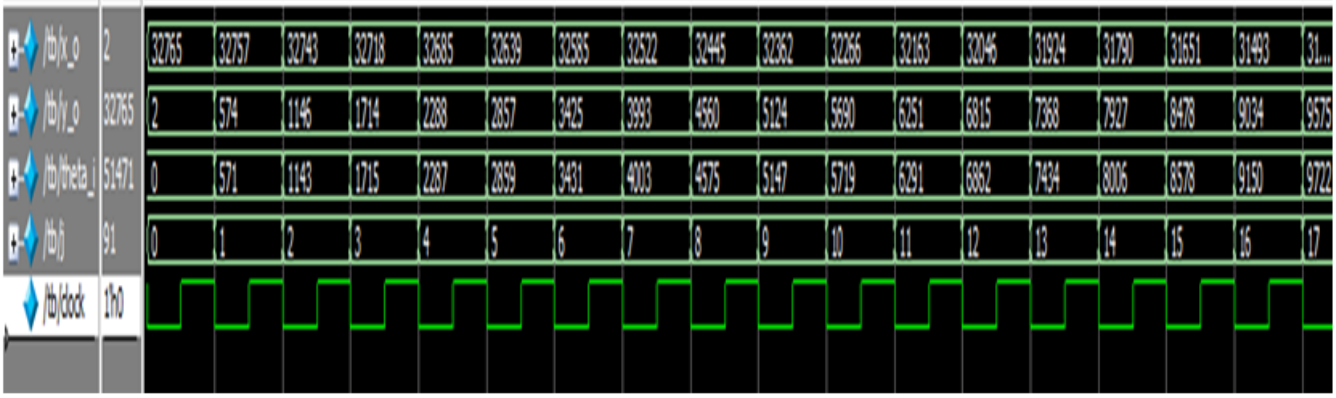

Fig 1. Parallel CORDIC Output Delay

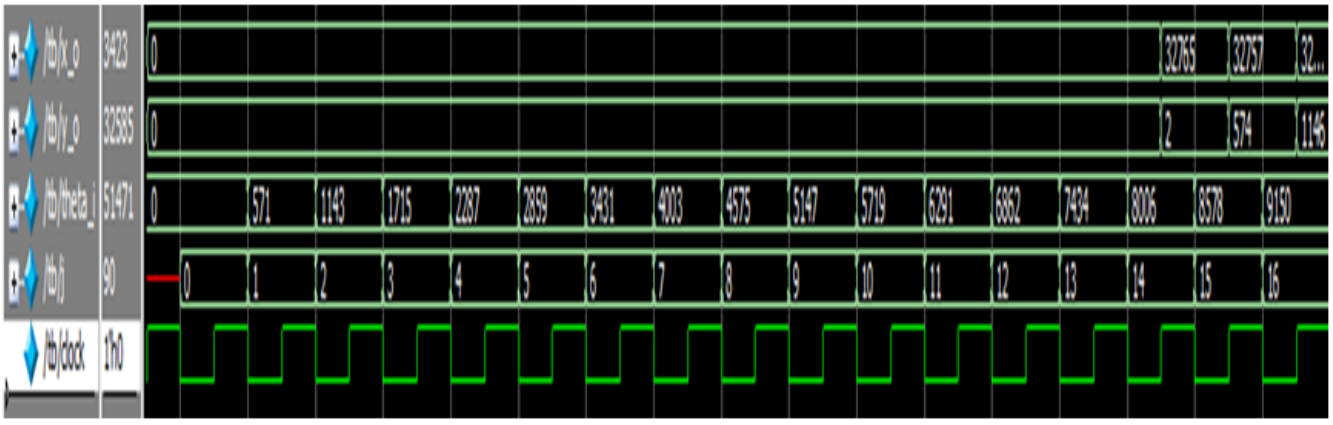

Fig 2. Pipelined CORDIC Output Delay

B. Total Area Comparison

If we compare the implemented area of the parallel CORDIC in the silicon with the pipelined CORDIC, we will find that the pipelined CORDIC will take bigger area than the parallel CORDIC.

C. Power Consumption Comparison

The power consumption of the pipelined CORDIC is more than the power consumption of the parallel CORDIC due to the area of pipelined CORDIC is bigger than parallel CORDIC.

D. Critical Path Delay Comparison

The critical path of the parallel CORDIC is from the input to the output of the CORDIC. In the pipelined CORDIC this path from the input to the output is divided into 14 paths due to adding pipelined flip-flops. The pipelined CORDIC critical path is smaller than the parallel CORDIC critical path.

E. Max Clock Frequency Comparison

The relation between the max operating clock frequency and the critical path delay is inverse relation. If the critical path delay is increased, the max operating clock frequency is decreased. As a result of this relation, pipelined CORDIC can work in higher clock frequencies than the parallel CORDIC.

V. RESULTS

Simulation of the pipelined CORDIC present to us extra initial delay appears from inserting the inputs till we get outputs from the

Pipelined CORDIC, however in the parallel CORDIC the outputs are valid in the next clock edge after inserting the inputs to the parallel CORDIC.

In table 1 we summarize our simulations comparison between the parallel CORDIC and the pipelined CORDIC. There are five comparison fields we use in our comparison delay, area, power consumption, critical path, clock frequency.

TABLE 1. Comparison between Parallel CORDIC and Pipelined CORDIC

Comparison fields	Parallel CORDIC	Pipelined CORDIC
Delay	One clock cycle	Fourteen clock cycle
Area	Small area as it consists of add and shift only	Large Area as it consists of add and shift and pipeline registers
Power consumption	Low power consumed due to small area	Large power consumed due to small area
Critical Path	Large critical path, it is the path between input to output	Small critical path, it is the path between two pipeline flip-flops
Clock frequency	Work in low clock frequency due to large critical path.	Work in low clock frequency due to large critical path.

VI. CONCLUSION

In this paper we make comparison between the pipelined CORDIC and the parallel CORDIC, as we see from the comparison the parallel CORDIC is more efficient in the initial delay, area, power consumption and has fast calculation speed, simple in the design, as it is pure combinational module. The parallel CORDIC has small area comparing to the pipelined CORDIC due to the parallel CORDIC is pure combinational. The pipelined CORDIC is more efficient in the work using higher clock frequencies because of small critical path.

REFERENCES

1. VOLDER, J. E. (2000). The Birth of CORDIC. Journal of VLSI Signal Processing , 101-105.
2. Takagi, Naofumi, Tohru Asada, and Shuzo Yajima., "Redundant CORDIC methods with a constant scale factor for sine and cosine computation", IEEE Transactions on Computers, vol.40, no.9, pp.989-995, 1991.
3. J. Chen and K. J. R. Liu, "A fully pipelined parallel CORDIC architecture for half-pel motion estimation," in Proc. IEEE Int. Conf. Image Processing, Santa Barbara, CA, Oct. 1997, vol. 2,pp. 574–577.
4. T. B. Juang and H. F. Lin, "CORDIC algorithm for vectoring mode
5. without constant scaling factors," Electron. Lett., vol. 35, no. 12, pp. 971–972, Jun. 10, 1999
6. T. B. Juang, "Area/delay efficient recoding methods for parallel CORDIC rotations," in Proc. IEEE Int. Symp. Circuits And Systems, June 2006, pp. 1539-1542.
7. G.Gopikiran, R.Thilagavathy, "FPGA Implementation of Floatingpoint Rotation Mode CORDIC Algorithm," " in Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011.
8. A. Syed, M. Lourde R., "Hardware Security Threats to DSP Applications in an IoT network," IEEE International Symposium on Nanoelectronic and Information Systems, 2016.
9. Aggarwal, Supriya, Pramod K. Meher, and KavitaKhare., "Concept, Design, and Implementation of Reconfigurable CORDIC", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.24, no.4, pp.1588-1592, 2016.