Design and Implementation Hardware Architecture for Four–Quadratic Arctangent

Omar Zeyad
University of Mosul

ABSTRACT
The Four-Quadratic Arctangent (atan2) function is used in many fields: telecommunication to recover the phase of the signal, robotics to computing the inverse kinematic for a robot arm control system, and in general used in transformation from Cartesian to polar coordinates. In this paper an FPGA based architecture design and its implementation has been presented. The piecewise polynomial approximation method with non-uniform segmentation has been used to implement the arctangent because it is suitable in hardware implementation. It is saving in resource utilization and takes a suitable accuracy.

Keywords
FPGA, Four–Quadratic Arctangent, piecewise polynomial approximation

1. INTRODUCTION
The atan2 is a function with two arguments(x, y), it is in fact the arctangent function with one argument (z) where z = y/x, but the signs of both arguments are used to determine the quadrant of the result as shown in equations [1].

$$atan(y,x)=\begin{cases}\mathbf{atan}\left(\frac{y}{x}\right), if\ x>0\\ \mathbf{atan}\left(\frac{y}{x}\right)+\pi, if\ x<0\ and\ y\geq 0\\ \mathbf{atan}\left(\frac{y}{x}\right)-\pi, if\ x<0\ and\ y<0\\ \frac{\pi}{2}\quad, if\ x=0\ and\ y>0\\ -\frac{\pi}{2}\quad, if\ x=0\ and\ y<0\\ undefined, if\ x\ and\ y=0\end{cases}$$

Because of the atan2 has been used in many fields therefore many research has been proposed to implement it. Florent de Dinechin and Matei Istoan are proposed an article that studies for this context the implementation of atan2 with fixed-point inputs and outputs. It compares the prevalent CORDIC shift-and-add algorithm to two multiplier-based techniques. The first one computes the bivariate atan2 function as the composition of two univariate functions: the reciprocal, and the arctangent, each evaluated using bipartite or polynomial approximation methods. The second technique directly uses piecewise bivariate polynomial approximations of degree 1 or 2. Each of these approaches requires a relevant argument reduction, which is also discussed. All the algorithms are last-bit accurate, and implemented with similar care in the open-source FloPoCo framework [2]. R. Gutierrez a, V. Torres b and J. Valls presents a paper that proposed an architecture for the computation of the atan2. Because of the throughput is between 20 and 40 MHz therefore it is suitable for broadband communications systems. The division operation of two inputs in the proposed architecture has been implemented by means of a logarithmic transformation, in which the division can be performed with a subtraction. The architecture has been proposed a technique that is uses a combination of non-uniform segmentation and multipartite LUT. A Xilinx FPGA device has been used to implement the architecture to achieve higher throughput than the approach based on CORDIC algorithm and lower area than previous LUT-based approaches [3]. Abhisek Ukil and etc. e present a research to implement the arctan function using look-up table, the look-up table is consist of 101 points and then the accuracy is increased by linear interpolation. The four quadrant operation has been computing used particular properties of arctan. The microcontroller platform with a 60MHz has been used to implement the proposed scheme because of it is typical for embedded applications. The accuracy of the proposed method is better than the reported approximation techniques. The speed of the proposed method is significantly more than the library function of the same microcontroller platform. [4].

2. IMPLEMENTATION
The Atan2 architecture is consist of three units. Figure 1 shows the block diagram of Atan2 architecture, the absolute value will be taken for the inputs (x and y), then the divider unit will used to divide them and the result is used as input to Atan unit. The Atan unit is used to calculate the inverse tangent from 0 to 262144.998046875. The comparator is used to check the x value, if it is zero then the output of Atan2 unit is π/2 or - π/2 depending on the sign of the y. The divider unit is implemented using the Xilinx LogiCORE™ IP Divider Generator v3.0 that creates a circuit for integer division based on Radix-2 non-restoring division. The Radix-2 algorithm exploits fabric to achieve a range of throughput options that includes single cycle [5].

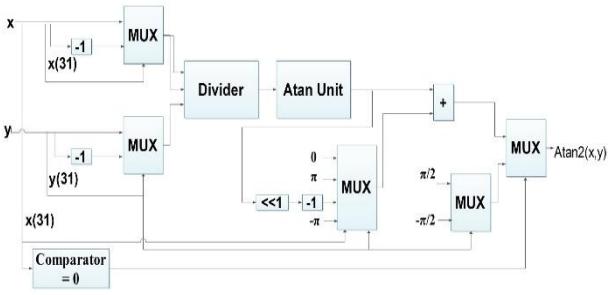

Fig 1: The block diagram of Atan2 architecture

The implementation of Atan unit is used the piecewise polynomial approximation method with non-uniform segmentation. The inverse tangent function has been segmented non-uniformly areas depending on linearity and non-linearity of the function. The segmentation process is depend on the non-linearity of the regions, the regions with high non-linearity is need small segments and the regions with small non-linearity is need large segments. Figure 2 show that how the inverse tangent function is uniformly segmented, it is started with high non-linearity and gradually become smaller non-linearity, therefore at the beginning of the function a large number of segments will have been taken (smaller segments regions) and progressively the number of segment will became more (larger segments regions). This

process will decreasing the size of look-up tables and increasing the accuracy. Because of the inverse tangent function is Symmetric with Respect to the Origin, therefore the negative inputs values can be calculate by take the absolute value of the input and multiply the output of by -1, therefore the right side from the function can be used[6].

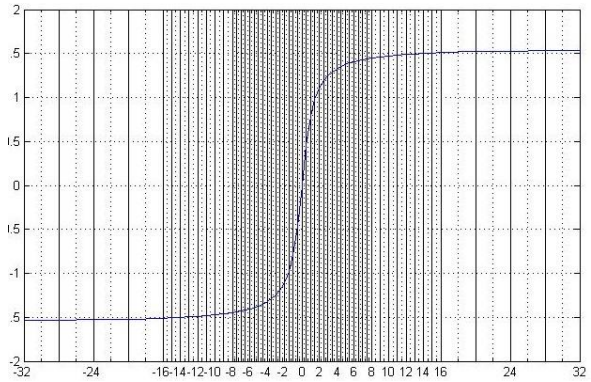

Fig 2: Atan Segmentation

The coefficients for each segment have been calculated using the minimax algorithm as shown in table 1 using maple software [7]. The regions from 0 to 8 had been segmented by 0.498046875 step, the length of the region depends on the number of fraction bits that represent the input, the input is represented by fixed point format as Q18.9. Atan(x) is equal to a1*x+a0.

Table 1. The coefficients of each segment

From	To	a0	a1
0	0.498046875	0.0066074118	0.9277919644
0.5	0.998046875	0.1510420675	0.6440619421
1	1.498046875	0.3962185158	0.3951315951
1.5	1.998046875	0.6127672027	0.2489003955
2	2.498046875	0.7762786564	0.1663932813
2.5	2.998046875	0.8975118787	0.1175800886
3	3.498046875	0.9889649206	0.08694646189
3.5	3.998046875	1.059657894	0.06667254635
4	4.498046875	1.115617192	0.05264116530
4.5	4.998046875	1.160861296	0.04256273588
5	5.498046875	1.198120050	0.03509617155
5.5	5.998046875	1.229292639	0.02941895478
6	6.498046875	1.255732464	0.02500603342
6.5	6.998046875	1.278425955	0.02151042632
7	7.498046875	1.298106982	0.01869582628
7.5	7.998046875	1.315331726	0.01639701574
8	8. .998046875	1.337033786	0.01370070886
9	9. .998046875	1.361365439	0.01099069327
10	10. .998046875	1.381130108	0.00901035158
11	11. .998046875	1.397498712	0.007519870550
12	12. .998046875	1.411274670	0.006370291906
13	13. .998046875	1.423027181	0.005465184955
14	14. .998046875	1.433170570	0.004739915389
15	15. .998046875	1.442013495	0.004149858339
16	19. .998046875	1.458876764	0.003115406985
20	23.998046875	1.479444277	0.002079123042
24	27.998046875	1.493604651	0.001485970118
28	31.998046875	1.503954606	0.001114887790
32	127.998046875	1.535650621	0.0002440401021
128	511.998046875	1.562007421	0.00001525843983
512	262144.998046875	1.569731539	0.00000000745054

The figure 3 shows the block diagram of Atan unit, the number of the regions that the Atan function segmented is 31 regions, therefore at least 32 * 27 bit lookup table is needed to store the a0 coefficient which is represented by 27 bits with fixed point format as Q18.9 and 31*31 bit is needed to store a1 coefficient which is represented by 31 bits with fixed point format as Q0.31. The first 15 bits are used as inputs for comparator that is encoded to create the selector of the multiplexer that select the address of the look-up tables, then the coefficients will read to calculate the output (atan) that consist of 18 bits (1 bit for integer part and 17 for fraction part).

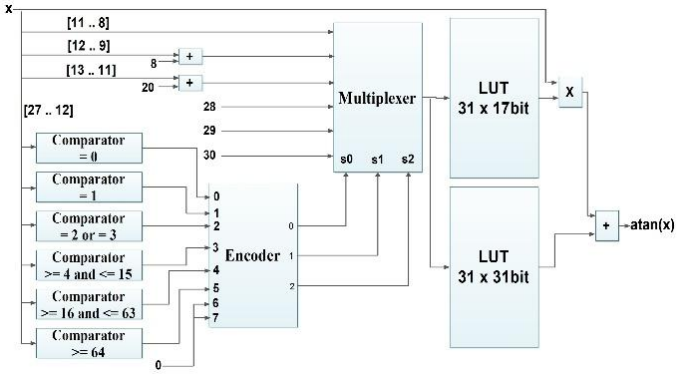

Fig 3: The block diagram of Atan unit

Three points have been chosen to test the architecture (p1(1,4) ,p3(0.5,0.25), p5(0.005,70.54)). Figure(4) show the simulation result of the three points that be chosen, As shown in figure the output(z) need to 49 clocks to get the first result and then after each clock the architecture will give a new result. The output bit (rdy) is use as ready bit when the output is ready to use. The table 2 show that the inputs points and the output results (theoretical and practical) and the error rate between the theoretical and practical results.

Table 2. The inputs points and the output results

Inputs		Output		Error %
X	Y	Theoretical	Practical	
1	4	1.325	1.326	0.075
0.5	0.25	0.463	0.473	2.16
0.005	70.54	1.570	1.569	0.063

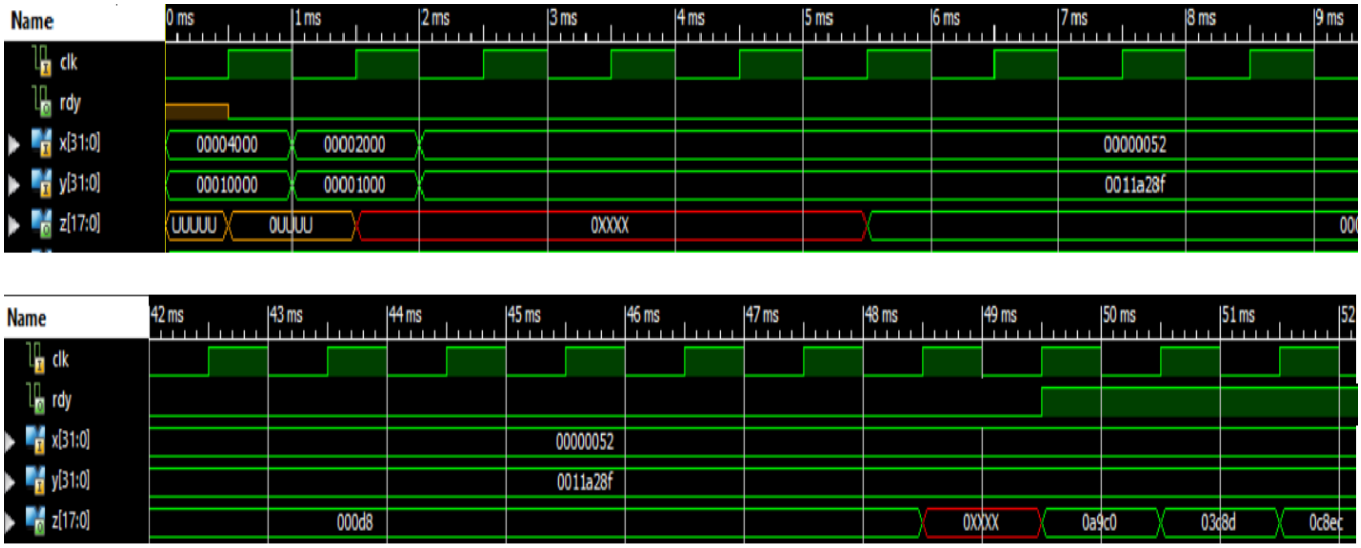

Fig 4: The simulation results of Atan2 architecture

3. RESULTS

The pricewise polynomial approximation method is depend on divide the function to segments and find the coefficients for each segments [8]. In this article the non-uniform segmentation has been used to divide the Atan function. The figure 5 show the error rate between the theoretical result of Atan and the pricewise polynomial approximation results, as figure shown the error is maximum in the non-linearity region of Atan function and gradually decreased to zero and then increased when the input equal to 32 because the segment region is large. The maximum error ratio is 2.62% and this error ratio is acceptable and non-influential therefore the pricewise polynomial approximation with non-uniform segmentation method is take a best accurate.

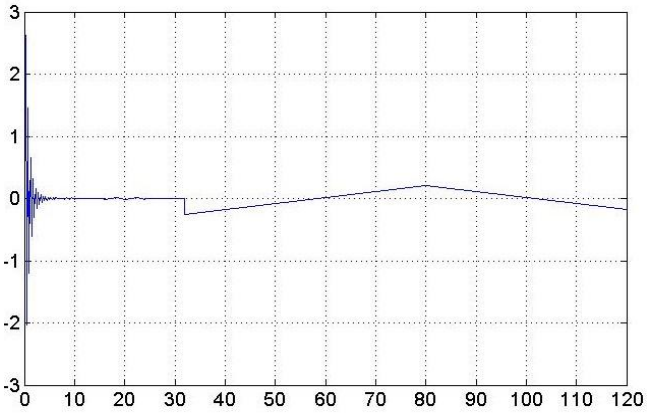

Fig 5: Error Rate

The target device that is used to implement the architecture is Xilinx Spartan 6 that has a XC6SLX16 FPGA chip [9]. Table 3 show the hardware utilization results, the resource consumption process is acceptable, the maximum frequency is 191.797MHz, the architecture take 260.5ns (50*5.21ns) to getting the first output.

Table 3. Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	4345	18224	23%
Number of Slice LUTs	1771	9112	19%
Number of fully used LUT-FF pairs	1545	4571	33%
Number of bonded IOBs	84	232	36%
Number of BUFG / BUFGCTRLs	1	16	6%
Number of DSP48A1s	1	32	3%

4. CONCLUTION

This article present a method to implement the Atan2 function using FPGA, this method is used non uniform segmentation of the function, and then the coefficients has been calculated for each segments. The number of segment is depend on the frequently of non-linearity of the function and the needed accuracy therefore many segments has been taken to get a very good accuracy, the accuracy can be increased by taking more segments but this will result increased the resource utilization or take less segments and this will result to decreased the resource utilization and accuracy. The presented architecture is suitable to implement using FPGA because of the hardware utilization and maximum frequency, it is consume a reasonable number of resources and take a high speed clock. The proposed architecture is suitable to be used to speed up the invers kinematic in robot arm because of the atan2 function is mainly used in parameters evaluation of invers kinematic.

5. REFERENCES

[1] Tildon H., 2011, Introduction to Circuit Analysis and Design, Springer Science & Business Media.

[2] Roberto Gutierrez and Javier Valls, 2009, Low-power fpga-implementation of atan (y/x) using look-up table methods for communication applications, Journal of Signal Processing Systems.

[3] Florent de Dinechin and Matei Istoan, 2015, Hardware Implementations of Fixed-Point Atan2, Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on.

[4] Abhisek Ukil and etc, 2011, Fast Computation of arctangent Functions for Embedded Applications: A Comparative Analysis, Industrial Electronics (ISIE), 2011 IEEE International Symposium on.

[5] LogiCORE IP Divider Generator v3.0, www.xilinx.com, DS530 March 1, 2011.

[6] Nelson H.F. Beebe, 2017, The Mathematical-Function Computation Handbook, Springer.

[7] "Maplesoft Online helps", http://www.maplesoft.com/support/help/

[8] Jean-Michel Muller, 2006, "Elementary Function Algorithms and Implementation", Second Edition, Birkhauser Boston.

[9] "Spartan-6 Family Overview", www.xilinx.com, DS160 (v2.0) October 25, 2011.