Document downloaded from:

http://hdl.handle.net/10251/108350

This paper must be cited as:

Torres Carot, V.; Valls Coquillat, J.; Canet Subiela, MJ. (2017). Optimised CORDIC-based atan2 computation for FPGA implementations. Electronics Letters. 53(19):1296-1298. doi:10.1049/el.2017.2090

The final publication is available at

http://doi.org/10.1049/el.2017.2090

Copyright Institution of Electrical Engineers

Additional Information

Optimized CORDIC-based atan2 computation for FPGA implementations

V. Torres, J. Valls and M.J. Canet

1st June 2017

Abstract

In the present article we describe a method for the implementation of the atan2 operator based on the CORDIC algorithm. In our proposal the computation of the z-path takes advantage of the LUT-based Field Programmable Gate Array (FPGA) resources to reduce by between 17% and 25%, without performance deterioration, the overall area of the unrolled architecture.

1 Introduction

The atan2(y,x) function is an operation frequently needed in hardware systems. For example, it is required to compute the angle of complex numbers in the time/frequency/phase synchronization stages of digital communication systems. Since the atan2 function is highly nonlinear and has two input variables, the overall implementation cost of the operator can be significant. In this regard, hardware-optimized operators are always desirable in real-world designs to reduce their economical cost or to improve the performance without increasing that cost. The iterative algorithm COordinate Rotation DIgital Computer (CORDIC) [1] in its vectoring mode is probably the best option for hardware implementations of the atan2(y,x), and the conventional version is the best choice for FPGAs because it profits from the fast carry propagation logic, which is faster than general logic [2]. In vectoring mode, the aim of the CORDIC algorithm is to rotate its input vector (x,y) trying to align it with the x axis to obtain the atan2(y,x) value in the z variable. The direction d_i of the rotation at iteration i is decided by the sign of y_i: $d_i{=}{-}sign(y_i)$. The initial values for the algorithm are $z_0{=}0$, $x_0{=}x$, $y_0{=}y$ and the CORDIC equations for iteration i are as follows:

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i}, \qquad y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}, \tag{1}$$

$$z_{i+1} = z_i - d_i \cdot atan(2^{-i}), \tag{2}$$

Since the range of convergence of the algorithm is $|z|{\leq}1.74329$, some sort of preprocessing, and may be also postprocessing, must be done to ensure the

convergence for any possible input vector. Without loss of generality, we will assume the use of the following preprocessing stage [1] that guarantees the convergence for any input vector because it moves the initial angle of the regular CORDIC to the $[-\pi/2, \pi/2]$ range: $x_0{=}{-}d_{-1}{\cdot}y,\ y_0{=}d_{-1}{\cdot}x,\ z_0{=}{-}d_{-1}{\cdot}\pi/2$, being $d_{-1} = sign(y)$. To have a w-bit output precision, $w{+}1$ iterations are needed.

In the present work we propose a method to reduce the cost of the z-path in the conventional CORDIC for FPGA devices.

2 Optimizations for the CORDIC atan2

Several optimizations can be applied when the only output of the algorithm is atan2(y,x): a) the width of the y-path can be progressively reduced, since $|y_i|$ has a bound that is divided by 2 with each iteration, b) the x-path can be frozen, that is, it is not updated any more as soon as this does not negatively affect the output accuracy, since the only output of the operator is z, c) the adder/subtractors in the x-path can have a shorter word length than those in the y-path, since this last data-path needs more precision to guarantee the right decisions and d) the y-path adder/subtractor for the final iteration can be omitted because its output is irrelevant for the atan2 computation. All the optimizations explained above can be applied and the accuracy goal at the output of the operator is still satisfied, which in this work we assume is last-bit accuracy (LBA), *i.e.* the maximum error magnitude is less or equal than the weight of the least significant bit (LSB) of the output. For example, the operator in Fig. 1 is a 12-bit CORDIC where the x-path is frozen after the 4th regular CORDIC stage and the sizes of the adder/subtractors are drawn in accordance to their real word length.

3 LUT-based computation of the z-path

In the conventional CORDIC, the angle z_{i+1} is computed from the angle z_i accumulated in the previous iterations and the angle rotated in the current iteration, according to (2). Therefore, the update of the z-path requires an adder/subtractor for each CORDIC iteration. In the present paper we propose a novel implementation scheme that profits from the resources offered by modern FPGA devices. These devices provide look-up tables (LUTs) connected to carry-chain logic to implement fast ripple-carry structures, as shown in Fig. 2a. These resources can be used to store precomputed values for all the possible rotated angles for n consecutive CORDIC iterations and, also, to add that value to the angle z_g accumulated in previous stages. Note that, despite the reduction in the use of resources that we will show in next sections, with this proposal we are still benefiting from the carry-chain logic, as shown in Fig. 2b. Hence, if the table that stores the precomputed values for n iterations is named LUTn, the new value for the angle is obtained as:

$$z_{g+n} = \mathrm{LUTn}(d_g, \ldots, d_{g+n-1}) + z_g. \tag{3}$$

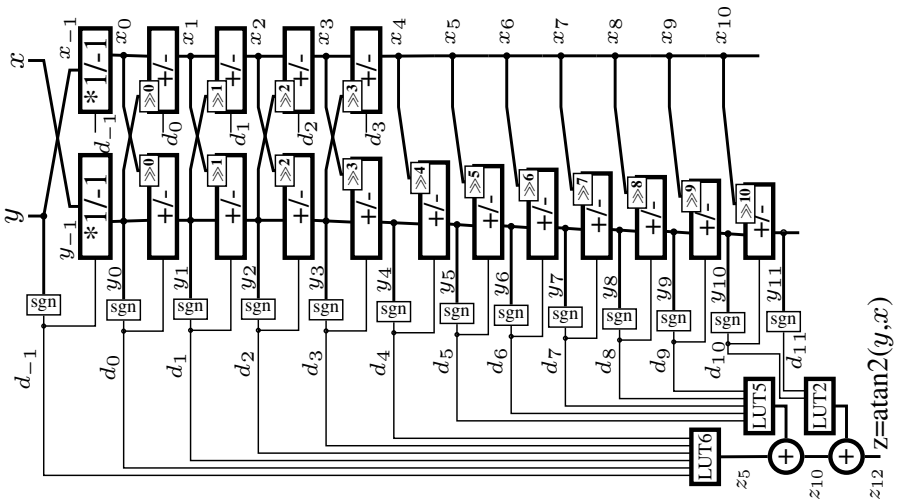

Figure 1: Schematic diagram of the operator. The first stage of the operator is a $\pm\pi/2$ rotator, as described in the text. The sgn block is an operator that extracts the sign bit of its input. LUTns blocks and one adder are implemented using 6-input LUTs and carry-logic resources.

Assuming that all the iterations follow equations (1)–(2), the content of the LUTn for a set of n stages is filled using the following expression:

$$\mathrm{LUTn}(d_g, \ldots, d_{g+n-1}) = -\sum_{i=g}^{g+n-1} d_i \cdot atan\left(2^{-i}\right). \tag{4}$$

Specifically, current Xilinx slices provide 5-input LUTs whose output can be added to another input, as shown in Fig. 2a. Therefore, the rotated angle for sets of $n \leq 5$ iterations can be stored using one 5-input LUT per bit of the angle. Since for the first 6 iterations there is no accumulated angle, the basic structure of the FPGA can be configured to act as a LUT6 without an adder. After that, as many LUT5 plus adder structures should be used as needed (*e.g.* only one of these structures is needed in Fig. 1). The LUTn for the final set of iterations may be smaller than in previous stages (*e.g.* a LUT2 is needed in Fig. 1). The required amount of LUTns for a w-bit precision CORDIC is:

$$n_s = 1 + \lceil (w{+}1{-}6)\,/5 \rceil. \tag{5}$$

For example, for a 24-bit CORDIC, instead of the 25 adder/subtractors required by a conventional z-path, only 5 LUTns are needed: one LUT6, three LUT5 plus adder, and a final LUT4 plus adder.

Finally, the output of the operator can be rounded to w fractional bits just by adding $2^{-(w+1)}$ to one of the LUTns that store the combinations of atan values and, then, simply truncating the output of the operator.

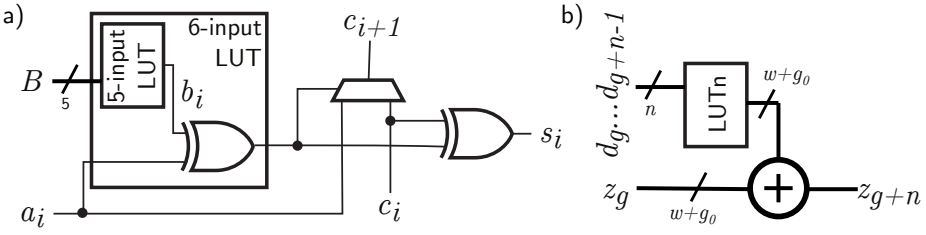

Figure 2: Basic blocks for the z-path implementation*a* Bit slice of a LUT and its connection to the carry-chain in Xilinx FPGAs. *b* For each set of n iterations, the updated value for the z-path is computed by adding the angle accumulated in previous iterations to the precomputed rotation for the current set of n iterations.

4 Cost reduction analysis

Since we propose a novel architecture for the computation of the z-path, without loss of generality we can assume that the x_i, y_i values are computed with enough accuracy to ensure that their rounding errors do not change the decisions in a way that cannot be corrected by CORDIC. We also assume that the output format is normalized to the $[0, 1)$ range with the same word-length w as the inputs.

Assuming that w+1 CORDIC stages are required to guarantee the desired output accuracy (*e.g.* LBA), and since the first two atan values used in the z-path are exactly rounded (*i.e.* their normalized values are 0.25 and 0.125), in a classical scheme $w-1$ rounded atan values are added, what gives a theoretical upper bound for the absolute value of the error e_z due to rounding operations at the operator output of:

$$|e_z| \leq (w-1) \cdot 2^{-(w+g+1)}, \tag{6}$$

where g is the number of guard bits. On the contrary, in our proposal the n_s LUTns store precomputed combinations of n atan values, and, therefore, the rounding errors are added only n_s times (see (5)), so the upper bound for the absolute value of the error in the z-path due to rounding operations is:

$$|e_{zo}| \leq n_s \cdot 2^{-(w+g_o+1)}, \tag{7}$$

where g_o is the amount of guard bits in our proposal. Equaling (6) and (7) it is deduced that in order for our proposal to have the same z-path error bound as in the classical scheme, it must be satisfied: $n_s \cdot 2^{-g_o} = (w-1) \cdot 2^{-g}$, or, equivalently, the expected reduction in the amount of guard bits is $g - g_o = log_2((w-1)/n_s)$. It can be shown, by assigning values to w, that the predicted reduction in the amount of guard bits is around 2 bits. Nevertheless, that is a prediction related to upper error bounds that in a real design may not be reached and, as a matter of fact, the maximum error obtained in our simulations (data not shown) reveals

Table 1: Implementation results in a xc7k325tffg900-2 FPGA. Used abbreviations: LUTs: 6-input look-up-tables, reg: pipelining registers, cyc: total latency in clock cycles, f: maximum clock frequency (MHz)

	Conventional CORDIC			Proposed (z-path reduction)		
w	LUTs	reg	cyc@f	LUTs	reg	cyc@f
16	759	16	1@50	570	16	1@49
	759	88	2@96	568	70	2@93
	759	151	3@133	568	131	3@137
	759	211	4@174	568	177	4@174
	759	314	5@253	568	271	5@254
	747	483	9@344	553	402	9@353
	751	930	17@595	563	741	17@571
24	1676	24	1@28	1237	24	1@28
	1676	126	2@54	1237	111	2@54
	1676	225	3@78	1237	195	3@79
	1676	318	4@107	1237	261	4@106
	1679	406	5@128	1237	319	5@126
	1654	1091	13@266	1227	882	13@278
	1660	2116	25@485	1227	1664	25@462

that g_0=g−1 produces a similar error in the z-path with both methods due to the actual combination of the rounding errors. Therefore, the expected overall difference Δ_{LUT} in the amount of 6-input LUTs required in the z-path is:

$$\Delta_{LUT} \approx n_s \cdot (w + g - 1) - (w - 1) \cdot (w + g). \tag{8}$$

The amount of registers for pipelined versions of the operator is also expected to be reduced. In this regard, it should be noted that a shift register of up to 32 bits can be mapped in a single 5-input LUT (the so called SRL32 primitive) in a Xilinx FPGA. Therefore, if the computation of the z-path is conveniently performed in the last stages of the pipeline, the total amount of required registers is significantly reduced. For example, for a fully-pipelined version, the expected difference is:

$$\Delta_{reg} \approx (n_s + 1) \cdot (w + g - 1) - (w - 1) \cdot (w + g). \tag{9}$$

For example, for w=$\{16, 24\}$ with g=$\{4, 5\}$, (8) predicts a saving of around $\{224, 527\}$ 6-input LUTs, while for a fully pipelined operator, (9) predicts a saving of $\{205, 499\}$ 1-bit registers. In next section we provide actual implementation results that are a bit lower than these predictions. This is due to the optimizations performed by the synthesis tool.

5 Implementation results

In Table 1 we show the implementation results for a Kintex7 xc7k325tffg900-2A device from Xilinx. All the operators were modelled using VHDL language and

dimensioned to achieve LBA using the optimizations explained above.

The results in Table 1 clearly show that using the proposed method for the computation of the z-path can significantly reduce the overall use of resources of the operator without degrading its speed. A 25% reduction in the total amount of 6-input LUTs and a variable reduction in the amount of registers (dependent on the pipelining degree) are achieved. In the reported implementation results, output registers are always included, even for the purely combinational (1 cycle) implementations.

Our proposal was also tested using Altera Stratix-V devices, where the best approach is the use of sets of 6 iterations, instead of 5 as was the case in Xilinx devices. For a 16-bit operator, ALUTs are reduced from 703 to 575, and for a 24-bit operator they are reduced from 1575 to 1327. Therefore, an approximated overall 17% reduction in the total amount of required ALUTs is achieved without affecting accuracy nor speed.

From the results in [2], it can be concluded that one of the best alternatives to the CORDIC algorithm for the computation of the atan2 is the recip-mult-atan method (RMAM). This method first computes $z{=}a/b$ by means of a LUT that stores $1/b$ and a multiplier and, then, the one-variable function $atan(z)$ is computed using another LUT. We implemented that operator, designed for LBA, in the Xilinx device specified above using the FloPoCo tool provided by the authors of [2]. For 16-bit inputs this operator uses bipartite tables and needs 1042 6-input LUTs or 755 6-input LUTs and 1 DSP48. Our proposal (see Table 1) achieves the same accuracy with just 570 6-input LUTs. For a 24-bit operator, the RMAM is based on a Horner scheme and requires 2437 6-input LUTs or 794 6-input LUTs and 6 DSP48, while our proposal requires 1237 6-input LUTs and no DSP48.

6 Conclusions

We propose a novel strategy for the computation in FPGAs of the atan2 function based on a CORDIC algorithm, which reduces the cost of the z-path by adapting its design to the resources of FPGA devices and still profiting from the fast carry propagation logic. The method has been verified using FPGA devices from Xilinx and Altera, obtaining a 25% and a 17% reduction in the amount of 6-input LUTs, respectively, compared to the conventional approach. If future FPGA devices provide bigger LUTs in their basic cells, the savings could be much bigger.

7 Acknowledgements

This work is funded by the Spanish Ministerio de Economía y Competitividad and FEDER under the grant TEC2015-70858-C2-2-R.

V. Torres, J. Valls and M.J. Canet (Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València, Spain)

E-mail: jvalls@eln.upv.es

References

[1] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, K. Maharatna, '50 years of CORDIC: Algorithms, architectures, and applications,' *Trans. Cir. Sys. Part I*, 2009, **56** (9), pp. 1893–1907, doi: 10.1109/TCSI.2009.2025803

[2] F. de Dinechin, M. Istoan, 'Hardware implementations of fixed-point atan2,' in *22nd Symp. on Computer Arithmetic*, Lyon, France, June 2015, pp. 34–40, doi: 10.1109/ARITH.2015.23