FPGA Implementation of the Curve Generator Algorithm for H/W Acceleration Applications

I. JIVET, B. DRAGOI
Applied Electronics Department
University "Politechnica" Timisoara
V Parvan 2, Timisoara
ROMANIA
ioan.jivet@etc.upt.ro, beniamin.dragoi@etc.upt.ro, http://www.etc.upt.ro

Abstract: - The paper presents results of the implementation of a nonparametric curve generator algorithm in FPGA as a test of concept for H/W acceleration solutions. It is shown that coordinate sequences as a curve representation when generated by the algorithm form a nonuniform sampled cosine/sine sequences of values that can be used in signal generation applications. A known draw back of the algorithm is the phase non uniformity of the samples. The paper presents solutions to fix the non uniformity by sample time delay compensation. A time sequence compensated extension of the algorithm is presented for use in digital sine and cosine signal generation applications. The VHDL description of the extended algorithm as proposed was simulated and synthesized in a Xilinx family FPGA showing low gate count. Finally the paper explores a second extension of the algorithm for random value trigonometric function computation. The performance results are compared with recent known coordinate transform solutions implemented in FPGA using the CORDIC algorithm as reported in literature.

Key-Words: - H/W acceleration, Jordan's curve generator algorithm, FPGA implementation, CORDIC algorithm.

1 Introduction

Many architectures of digital systems in use today incorporate classical and new algorithms translated to hardware for speed and performance enhancement – hardware acceleration solutions. The CORDIC algorithm is the classical example most frequently used for the generation of trigonometric functions sequences [1].

FPGA implementation constitute the first target technology to explore on the feasibility and performance of algorithms translated to hardware [2].

The paper presents the results of a study exploring usability of the Jordan's nonparametric curve generator algorithm in areas of applications different from graphics [3]. The algorithm generates coordinates of points that 'follow' a line or quadratic curve in the plane. The basic principle of the algorithm is the embedding of a curve on a predefined point grid model of the coordinate space. Tracking it generates approximate trigonometric functions values.

One known shortcoming of the algorithm is that the tracking speed of the approximating point on the curve is not constant. This is one of the reasons for which the algorithm was not used in applications.

In applications where the stepping uniformity of the parametric description can be relaxed we show that the nonparametric generation algorithm with appropriate timing adjustments can be successfully used in practical applications. In the case the generated curve is a circle the coordinate pairs values form a sequence of sine and cosine values. These values can be used successfully in sinusoidal signal synthesis circuits.

In part 2 of the paper the nonlinearity of the Jordan's algorithm is analyzed and solutions for appropriate adjustments are suggested to emulate uniform parametric behavior for use in application of sinusoidal functions sequences generation.

Part 3 of the paper presents a VHDL description and synthesis results for the extended algorithm that proves suitable for sinusoidal signal generation as well as singular sine/cosine function values.

In part 4 of the paper a second extension of the algorithm form random sine/cosine values as used in coordinate transform applications. Performance capacity solutions are presented.

Part 5 of the paper presents conclusions and an outline of further development.

2 Algorithmic generation of signals

The principle of the Jordan's algorithm is a method of coordinate pairs generation by moving in the coordinate space from one grid point to another under the minimizing criteria of the distance to the curve.

It is easily visible that the segments of horizontal and vertical lines of the approximating path make successive angles with the tangent to the curve that vary along the curve. Using uniform stepping frequency along the path tracking the curve results in a variable speed of the phase of the rotating vector on the curve.

The values of the x and y coordinates pairs generated do represent a sequence of approximating sine/cosine values for the curve characteristic function but are non uniformly distributed with respect to the central angle steps.

If the sequences of values determined are used to generate the projections of the circle in sine and a cosine functions by presenting the values to an output at constant intervals errors in phase will occur. The resulting generated functions will have less then ideal phase and frequency spectrum characteristics.

If the application does not by its nature impose absolute uniformity of the distribution of the approximating samples, the results of our investigations show that the nonparametric algorithm method can constitute a viable alternative to the parametric method for the generation of trigonometric functions sequences.

According to the algorithm the coordinates of the next approximating point to the curve are obtained by moving to the adjacent grid point in either x or y direction. The simultaneous move on both coordinates is a variant of the algorithm very useful in graphics applications but not appropriate for signal generation application since it further complicates the non uniformity in the phase.

The decision of the axis to take the step on corresponds to the minimum absolute value of the implicit function at the two possible candidates. The sign of the increment on one axis is the sign of the opposite axis partial variable derivative of the circle implicit function.

The VHDL code presented in the following outlines the calculus of the coordinates pairs based on the partial derivatives and implicit function values necessary for the decision of the step direction at each current iteration following the original algorithm [4] [5].

-- VHDL outline - Curve Generator Algorithm

-- **Init:** flast = 0 - implicit function value; dfx = -1
-- dfy = +1, directional derivatives in 1st quadrant;
-- x = 2R, y = 0, start point on the circle,
-- xmid, ymid= 0, temporary variables.

```
begin
-- for i in 0 to 1000 loop
if clk'event and clk='1' then
-- set step direction inc parameters,
   --set y "00"on 0,"01"on+1,"10"on-1;
case dfx(14) is
      when '0' => y:="01";
      when '1' => y:="10";
      when others => null;
end case;
      -- case dfy(14) .... similar
-- recalculate implicit function values
case x is
      when "01" => fx := flast + 1 +
        (xmid(13 downto 0) & "0");
      when "10" => fy := flast + 1 -
         (xmid(13 downto 0) & "0");
      when others => null;
end case;
      --case y .... similar
-- decide step axis direction
if abs(fx) <= abs(fy) then
      dy:="00";  flast := fx;
      else dx:="00";  flast := fy;
end if;
-- update one position coordinate
case x is
      when "01" => xmid:= xmid + 1;
      when "10" => xmid:= xmid - 1;
      when "00" => xmid:= xmid;
      when others => null;
end case;
      --case y .... similar
-- update partial derivatives
case x is
      when"01"=> dfx := xmid(13
      downto 0) & "0" + 2;
      when "10" => dfx := xmid(13
      downto 0) & "0" - 2;
      when "00" => dfx := xmid(13
      downto 0) & "0";
      when others => null;
end case;
      --case y .... similar
--update,output step cycle coordinates
      x= xmid, xout <= x;
      y= ymid, yout <= y;
end loop.
```

Given the symmetry of the sine and cosine functions on the two axis results that the core part of the algorithm needs to be calculated for the first octant only and sign changed variants for the rest.

As it can be seen from the formulas used in the description only one comparison, two additions and several increments are to be executed at each step.

As presented in the original paper the non uniformity of the curve tracking given a constant stepping rate on the grid with high density is described by the formula:

$$Ni/R = 1 + \sin(sigma) - \cos(sigma) \qquad (1)$$

where Ni is the number of steps taken up to the i-th node, R the radius of the circle generated and sigma the phase angle. The formula indicates interleaved contribution from steps in both x and y directions.

Starting from the formula above it is easy to observe that the phase angle advancement at each step on X axis is proportional to the cosine of the current phase angle. By symmetry each step on Y axis is proportional to the sine of the phase angle.

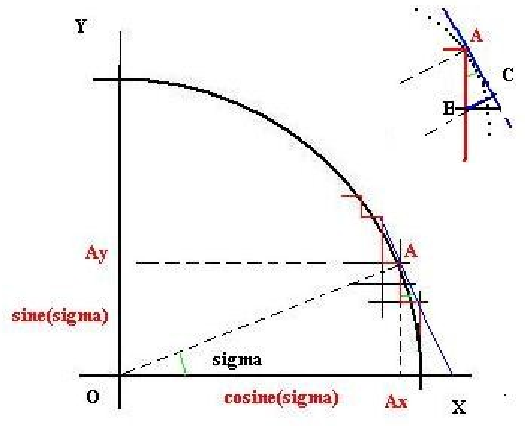


Fig.1 Description of projected stepping non uniformity along the curve and outline of the compensation method.

The observation can be used to extend the algorithm for generating sinusoidal sample sequences with appropriate timing to minimize phase distortion.

3 Extended algorithm with phase compensation

A geometrical derivation of the compensation method is presented below with reference to Fig. 1. The phase angle advanced at each step of the algorithm along one axis is proportional to the corresponding current phase sine or the cosine value as follows:

$$AC = AB \cos(sigma) \qquad (2)$$
$$\cos(sigma) = Ax/R \qquad (3)$$
$$AC \sim Ax \qquad (4)$$

The proposed extension of the algorithm for use in sine and cosine sequence generation with a correction in the phase is the following:

```
Loop:
    a)Select the x or y direction for the next step and the coordinate values according to the original algorithm;
    b)Take the generated coordinates , as sampled value sine and cosine functions;
    c)Determine the phase angle advancement along the curve for the current step;
    d)Present to the output the samples for each axis at a sample timing delay proportional to phase advancement;
    e)For a interleaved step on the opposite axis add corresponding delay for the phase advancement only and keep the sample value constant.
End Loop;
```

For ideal phase accuracy the best choice is to calculate and store the phase angle advancement at each step in advance. At signal generation the stored values will be used to time the samples to the output. The time non uniformity of the values generated will still be there but the phase of the generated signal will have no distortions.

A more practical solution that avoids the ROM hardware overhead is to use the just calculated coordinate value as an immediate approximation of the phase angle advancement. The coordinate values are really the current angle sine and cosine projection of the circle radius on the two axes.

With appropriate timing these samples can be used to generate accurate sine/cosine signals with considerable less resources in comparison to than other solutions reported in the literature [7], [8].

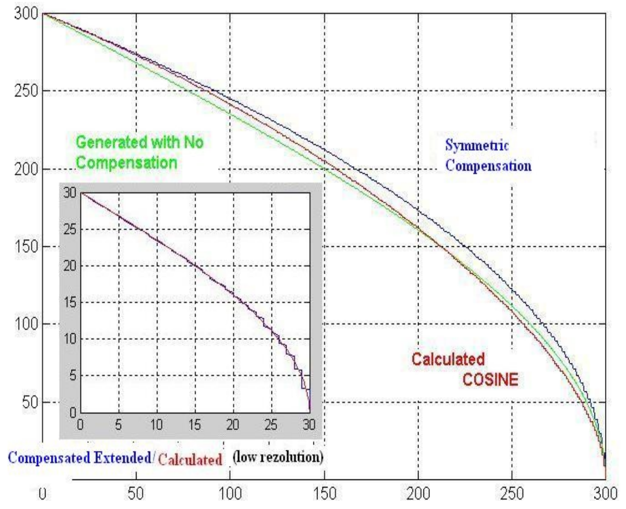


Fig 2. MathLab plot of algorithm generated cosine function, with compensation (with a low resolution insert as a close up) vs. a calculated cosine.

An incremental algorithm is thus proposed that involves only additions, compare and increments well suited for digital implementation.

A MathLad simulation of the generated signal obtained using the proposed extended algorithm was compared with a true calculated cosine function as a validation before the VHDL synthesis in FPGA. The plots represented in Fig. 2. shows that appropriate compensation is achieved using the extended algorithm as proposed.

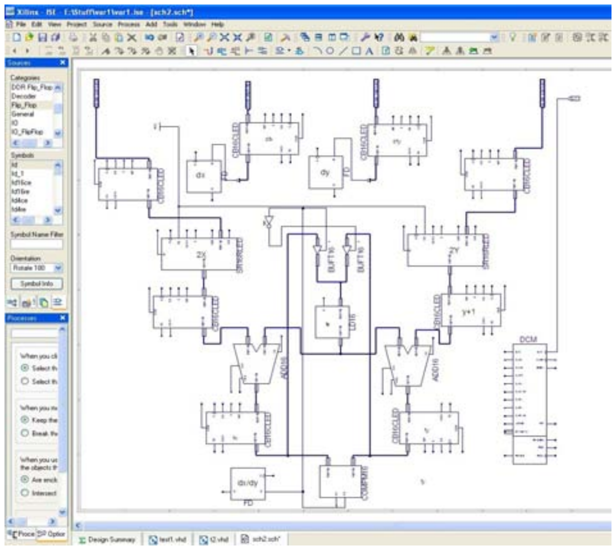


Fig. 3 Schematic of a manually instantiated implementation of the algorithm for Xilinx Vertex II.

The VHDL behavioral description of the algorithm was simulated and synthesized using ModelSim in a Xilinx ISE development environment.

The FPGA implementation as a test technology proved to be a valuable tool for real applications solutions exploration.

It was also found that the synthesis will yield better results if the functional units are instantiated and placed manually. The sequential generation of sine/cosine function values using the Jordan's algorithm was found not to be a limiting factor in its general use in other applications like random sine/cosine values determination in a minimum time slot.

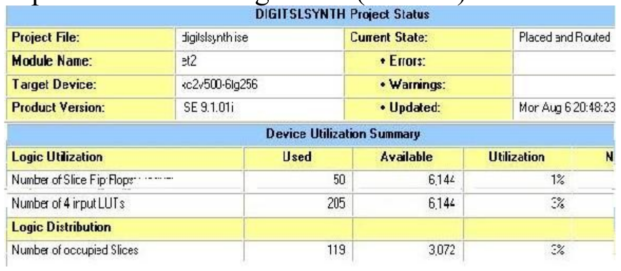
The class of applications requiring specific random sine/cosine values can thus receive solutions using the proposed extended algorithm as well.

Accelerated algorithm implementations can preserve the economy of the implementation of the algorithm satisfying at the same time the speed of generation found in classical trigonometric function values determination like the CORDIC algorithm.

Optimizing the circuit for minimum latency or for minimum resources is necessary for each application to fulfill the main set objectives.

Table 1

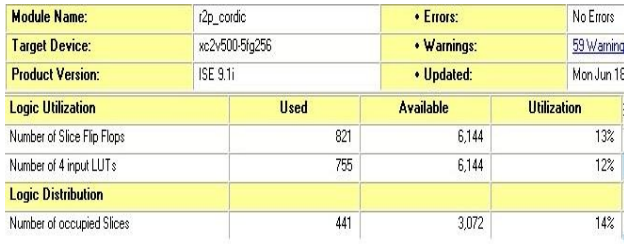
Proposed Extended Algorithm (16 - bit)



DIGITSLSYNTH Project Status			
Project File:	digitslsynth ise	Current State:	Placed and Routed
Module Name:	et2	• Errors:	
Target Device:	xc2v500-6lg256	• Warnings:	
Product Version:	SE 9.1.01i	• Updated:	Mon Aug 6 20:48:23

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	N
Number of Slice Flip Flops	50	6,144	1%	
Number of 4 input LUTs	205	6,144	3%	
Logic Distribution				
Number of occupied Slices	119	3,072	3%	

OpenCore CORDIC Core Synthesis (16-bit)



Module Name:	r2p_cordic	• Errors:	No Errors
Target Device:	xc2v500-5fg256	• Warnings:	59 Warning
Product Version:	ISE 9.1i	• Updated:	Mon Jun 18

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	821	6,144	13%
Number of 4 input LUTs	755	6,144	12%
Logic Distribution			
Number of occupied Slices	441	3,072	14%

The proposed extended algorithm and a FPGA Open Core CORDIC implementation where synthesized using the Xilinx, ISE 9.1 development environment software package [6], [12].

As shown in Table 1 the FPGA resources needed for the implementation confirms the practical technological efficiency for the use of the algorithm in sine/cosine generation applications [9], [11], [13].

Our synthesis results are similar to synthesis results reported by the Open Core [10], [12]. The low resource utilization is very encouraging and can be further reduced by optimization in the synthesis process.

Table 2
CORDIC Synthesis results for 16 bit Rectangular to Polar Transform compared to proposed method

Reference	Device	Slices/ cells	Max Clock
AlteraACEX	EP1K50-1	2190l cells	68MHz
XilinxSpartan	XC2S100-6	704 slices	93MHz
Algorithmic	XC2v500	119	10MHz

A factor of 4 to 10 reduction in resources cost per function is observed. The maxim complete computations per second ratio however is on the same order of magnitude in favor of the CORDIC solution.

4 Using the Algorithm in Coordinate Transform Applications

The range of applications using trigonometric functions values is a lot larger then signal generation. Single value computations are used in many areas of applications like coordinates transforms and discrete cosine transforms. The parameter of importance in these applications is the total sample computation time.

For the CORDIC algorithm sample computation time is independent of angle value and proportional to resolution **N**.

Using the extended version of Jordan's algorithm as proposed results in variable sample computation time with the angle value at a maximum worst case value of 2^N.

In order to use the extended algorithm time efficiently in random angle computation applications solutions to reduce the total calculation time must be developed.

A double resolution method is described in the following. The circle tracking algorithmic method is preserved and no major modifications are necessary to the hardware implementation.

The input parameters for the algorithm are: *the radius and the target phase angle*. The algorithm will return the values of the phase angle coordinates that correspond to radius scaled sine and cosine values. Therefore the algorithm acts as a *polar to rectangular coordinates transform.*

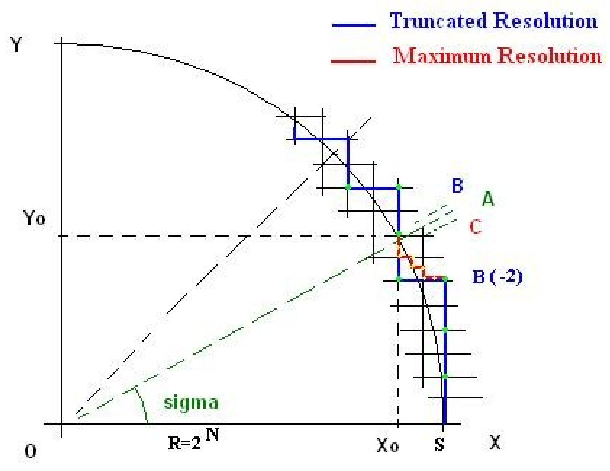


Fig 4. Example of coarse and fine stepping for the acceleration of the single value function calculations.

The basic idea of the method is to use a coarse resolution with large phase angle steps to quickly approach the target phase. The last steps are made after switching to maxim resolution.

The double resolution example of the use of the algorithm is given in connection with Fig. 4.

For a target phase angle an initially a coarse resolution is used (the path to point B). To continue the circle tracking the last few steps before a circle crossing must be back tracked to point B(-2) in this example. This is necessary to ensure that the algorithm will continue as if it arrived to this neighborhood via a fine stepping route of maximum resolution.

Changing the resolution is a simple repetitive shift operation of all the registers with the latest variables of the algorithm until maximum resolution is obtained.

Sample calculations show a significant reduction in calculation time when using this method.

Further work is needed to reduce the computation time to the order of resolution **N.** Also preprocessor architectures need to be devised to balance direct and inverse calculation methods and times. Novel FPGA fabric may be of great usefulness in the future work outlined.

5 Conclusions

An extension the Jordan's nonparametric curve generation algorithm is derived for use in sinusoidal signal generation applications. The extension of the algorithm given is such that the inherent non uniformity of the approximating points do not introduce phase errors.

The sample timing compensation based on current approximating point phase sine/cosine values is proposed. Simulation results indicate the adequacy of the solution as proposed.

The stepping intervals can be pre computed for a maximum accuracy of the generated sequence. An alternative is presented where the sample timing is proportional at each step to the just generated coordinate values for the step. A trade-off on the maximum generated frequency versus the phase accuracy must be made.

The efficiency of the implementation in FPGA is determined from a VHDL description synthesis and compared with similar precision CORDIC algorithm implementation as reported in the literature. The gate count efficiency of the proposed algorithm is shown to be a good trade-off for high frequency output sample values.

The algorithm implementation results are compared with recent known coordinate transforms solutions implemented in FPGA using CORDIC.

A second extension of the algorithm for random value trigonometric function values computation is also presented.

The exponential dependence of the calculation time as a function of the number of precision bits is shown to be a shortcoming of the algorithm. A solution is given to reduce this time by using coarse and fine resolutions

Further research is necessary to find solutions to reduce the calculation time gap for the extended algorithm to reach speed performance of CORDIC like implementations for trigonometric functions calculations.

References:

[1] R. Andraka, A survey of the CORDIC algorithms for the FPGA computers, *Proceedings of the ACM /SIGDA sixth international symposium on FPGA*, 3/1998, pp 191-200.

[2] T. C. Huang, H. T. Ho, Digital sine/cosine wave generator,*US Patent 6892213,* Issued May 10, 2005.

[3] F. Dengwei, and A. N. Willson, Two-Stage Angle-Rotation Architecture and Its Error Analysis for Efficient Digital Mixer Implementation, *IEEE Transactions on Circuits and Systems*, Vol.53, No. 3, March 2006.

[4]V. Boyer, Metaprogramming for the Generation of Nonparametric Curves, *EUROGRAPHICS*, 2001.

[5] B.W. Jordan, W.J. Lennon, and B.D Holm, An Improved Algorithm for the Generation of Nonparametric Curves, *IEEE Transactions on Computers*, Vol.C-22, No. 12, 1973, pp. 1052-1060.

[6] T. Vladimirova, H. Tiggeler, FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm, *Proceedings of 2006 MAPLD International Conference*, Washington D.C., Sept.26-28, 2006.

[7] D. Nowrouzezahrai, B. Decker and W. Bishop High-Performance Double-Precision Cosine Generation, *Proceedings of the 2005 International Conference on Computer Design*, Las Vegas, Nevada, USA, June 27-30, 2005 pp.42 – 48

[8] A. Madisetti, A. Kwentus, A.N. Jr. Willson, A Sine/Cosine Direct Digital Frequency Synthesizer using an Angle Rotation Algorithm, *1995 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, Feb 1995 pp:262 – 263.

[9] Numerically Controlled Oscillators, *MegaCore Function Users Guide,* www.altera.com/literature/ ug/*ug_nco.pdf* , May 07.

[10] CoreCORDIC CORDIC RTL Generator, *www.actel.com/ipdocs/CoreCORDIC_DS.pdf,* 2006.

[11]J. Duprat and J. Muller. The CORDIC algorithm: New results for fast VLSI implementation. *IEEE Transactions on Computers*, 42(2), Feb. 1993, pp:168–178.

[12] CORDIC Core, http://www.opencores.org/ projects.cgi/web/cordic, OpenCores, Feb 2004.

[13] NCO Megacore Function/User Guide, http://www.altera.com/literature/ug/ug_nco.pdf, ALTERA, May 2007