Area and Power Optimization of Chaotic Oscillator Using RK4 Method

Pavan Kumar R[1]
Department of ECE, CMRIT
Visvesvaraya Technological University
Belgaum, Karnataka
pavankrnaik@gmail.com

S. Shridevi[2]
Asst. prof. Department of ECE, CMRIT
Visvesvaraya Technological University
Belgaum, Karnataka
sridevi.s@cmrit.ac.in

Abstract— **Chaos theory studies the behavior of dynamical systems that are highly sensitive to initial conditions. Discretization of such systems can completely change the dynamics of the system even destroying the chaotic behavior. Fourth order Runge - Kutta algorithm is used here to design the chaotic oscillator. In this paper the clock division, floating point adder and field function generation is implemented in Verilog. Stability analysis of the system suggests the range of step size values. IEEE 754 standard floating point arithmetic is implemented with some normalization steps.**

The simulation is done by using Xilinx 14.5 software tool. The clock decision is mainly depends on the algorithm used in this design.

Keywords-- **Chaos theory, Runge-Kutta, Verilog, Xilinx**

I. INTRODUCTION

The chaotic systems are deterministic because they are governed by some mathematical model. The chaotic oscillators mainly depend on initial conditions and step size.Initial conditions effect is popularly referred as the Butterfly effect .The step size parameter h will be used in design algorithm in further part of the project. The design process is subdivided into 3 parts.

- Clock division and floating point adder
- Field function generation
- Slope generation

First part is done with simulation outputs. The clock is initially divided into 34 clock cycles. Fourth order algorithm suggests getting 136 clock cycles. The 16-bit floating point arithmetic is implemented which first normalizes the number to perform addition. Some precision will be lost because of normalization process. Runge-Kutta formulas are among the oldest and best understood schemes in numerical analysis. However, despite the evolution of a vast and comprehensive body of knowledge, it continues to be a source of active research. Runge-Kutta methods provide a popular way to solve the initial value problem for a system of ordinary differential equations.The construction of higher order ones is very complicated and many extra stages have to be inserted for achieving accuracy of the same magnitude with the basic formula.Such accuracy is useful in astronomical applications like LISA program which is space mission to be launched.

II. RUNGE–KUTTA METHOD

Euler’s method and the improved Euler’s method are the simplest examples of a whole family of numerical methods to approximate the solutions of differential equations called Runge-Kutta methods.

The trickiest part of using Runge-Kutta methods to approximate the solution of a differential equation is choosing the right step-size. Too large a step-size and the error is too large and the approximation is inaccurate. Too small a step-size and the process will take too long and possibly have too much round of error to be accurate. Furthermore, the appropriate step-size may change during the course of a single problem. Many problems in celestial mechanics, chemical reaction kinematics, and other areas have long periods of time where nothing much is happening (and for which large step-sizes are appropriate) mixed in with periods of intense activity where a small step-size is vital. What we need is an algorithm which includes a method for choosing the appropriate step-size at each step.

The Runge-Kutta-Fehlberg methods do just this. There are many different Runge-Kutta-Fehlberg methods, all of which involve comparing two different Runge-Kutta approximations to get an estimated error and step-size. The RK2(3) method is a low order method which gives moderately accurate results with only 3 function evaluations at each step. If high accuracy is important, you should use a higher order method. On the other hand, if you just need an answer to within 1% for a problem that doesn’t cover too large a range of x values, the RK2(3) method is quick, cheap and effective (with the recommended tolerance of T = .001).

The RK4 discrete system for the continuous system given by Eq. 1 is, [5]:

$$x(n+1)=x(n)+\frac{1}{6}\left[k_x^1(n)+2k_x^2(n)+2k_x^3(n)+k_x^4(n)\right]$$

$$y(n+1)=y(n)+\frac{1}{6}\left[k_y^1(n)+2k_y^2(n)+2k_y^3(n)+k_y^4(n)\right]$$

$$z(n+1)=z(n)+\frac{1}{6}\left[k_z^1(n)+2k_z^2(n)+2k_z^3(n)+k_z^4(n)\right] \qquad (1)$$

Wherex(n),y(n), and z(n) are the time series that constitutes the output of the project, and the values of $k_{\cdot}^{n}$ for i = 1, ..., 4 are:

$$k_x^1(n)=h\,f_x\left[x(n),y(n),z(n)\right]$$

$$k_x^2(n)=h\,f_x\left[x(n)+\frac{k_x^1}{2},y(n)+\frac{k_y^1(n)}{2},z(n)+\frac{k_z^1(n)}{2}\right]$$

$$k_x^3(n)=h\,f_x\left[x(n)+\frac{k_x^2}{2},y(n)+\frac{k_y^2(n)}{2},z(n)+\frac{k_z^2(n)}{2}\right]$$

$$k_x^4(n)=h\,f_x\left[x(n)+k_x^3(n),y(n)+k_y^3(n),z(n)+k_z^3(n)\right]$$

Where h is the step size. Similar expressions are obtained for y and z.

III. STABILITY OF RUNGE-KUTTA METHOD

It is known that RK methods with fixed step size h are stable if hλi ∈ D for all Eigen values λi of the local linearization, where D is a domain that can be calculated from the particular method employed. Specifically, for the classical fourth-order method (RK4) applied to a system with real Eigen values; the stability requirement is essentially that hλi ∈ (−2.79, 0). Therefore, the largest magnitude eigenvalue limits the size of the largest stable step size. Our linearized system has eigenvalues of λ1 = −2 cos (φ), λ2 = 0. Assuming we are near the steady state solution, sin (φ) ≈ .25 yields λ1 ≈ −1.93. Therefore, we expect h < 1.44 for stability. The stability polynomial is

$$R(h) = 1 + hb^T(I - hA)^{-1} \qquad (2)$$

And it is required thatR(h)<1 for absolute stability. The stability region is plotted using Map as follows

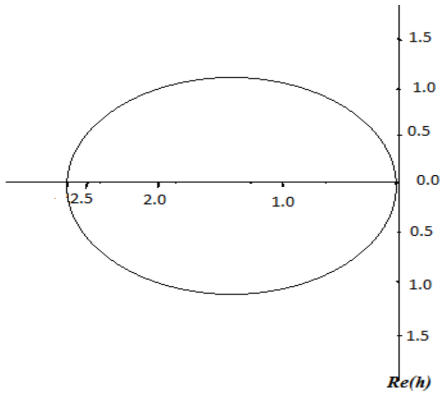

Fig.1 Stability graph of fourth order Runge-Kutta algorithm

IV. ARCHITECTURE

The process starts when the initial conditions for the state variables {x0, y0, z0} are loaded in the Register. Values are simultaneously sent to the first and second adder. The block k- generator is loaded with a zero value.

The block Field function produces the value to which are available at the input of the block Slope generator.
The block Field function requires at least 34 clock cycles of the main clock (clk) to finish all the calculations and give a valid result for f•. That is the reason c0 = clk/34. Furthermore four cycles of c0 are required to get the new value of the time series. Consequently the overall frequency turns out to be c1 = c0/4 = clk/136.

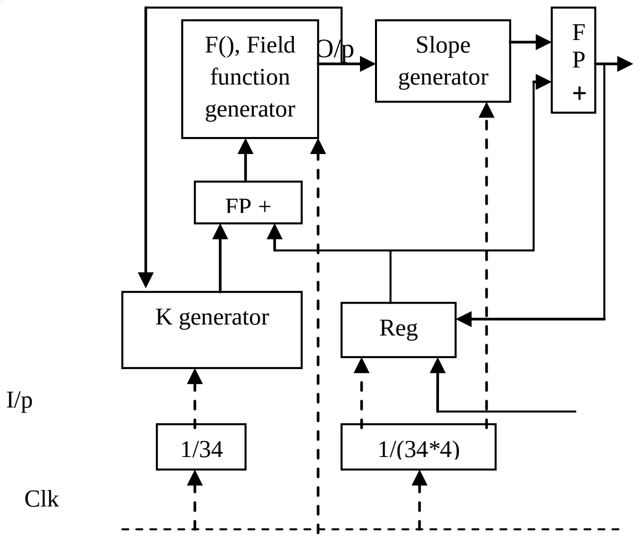

Fig.2 fourth order Runge-Kutta oscillator architecture

This block produces the output sent to the second Adder to obtain the new value of x1. Identical blocks are used for variables y and z. Once x1, y1and z1 are known they are sent to the block Register andthe cycle starts again. Blocks k-generator and Field-function constitute the brain of the RK4 solver and are reused as explained above to reduce the required FPGA area and power.

V. RESULTS

It is well known that chaotic systems are very sensitive to initial conditions. Consequently it is not possible to compare different realizations of the same system by means of the respective time series. It was shown that a statistical evaluation of the respective time series is a better methodology [7, 6, 4].The systematic statisticalevaluationofdifferentrealizationsinFPGAisawork in progress, but preliminary results show that the implementation in RK4 presented here may be used to produce chaotic time series with the same statistical properties than those obtained by numerical integration with more involved variable step methods, up to h = 0.05. Thisvalueofhrepresentsanimportantimprovement over our previous realization using the Euler algorithm that required h $\leq$ 0.0045 [5]. These 10 times higher h implies that the number of time intervals required to cover a given evolution time is reduced in a factor 10 and then it reduces the FPGA memory bits required.

.

VI. CONCLUSION

A method to real time implementation of a dynamical chaotic system onto a FPGA board is introduced. In free resources it is possible an improvement of the proposed design. Future work will include the use of pipelining, parallel computation and variable step algorithm.

REFERENCES

[1] Ch. Tsitouras "Runge-Kutta interpolants for high precision computations"
[2] Joseph D. Skufca "Analysis Still Matters: A Surprising Instance of Failure of Runge–Kutta–Felberg ODE Solvers" Vol. 46, No. 4, pp. 729–737.
[3] H. Musa, Ibrahim Saidu and M.Y. Waziri "A Simplified Derivation and Analysis of Fourth Order Runge Kutta Method" Volume 9– No.8, November 2010.
[4] L. DeMicco, O. G. Zabaleta, C. M. Gonzlez, C. M. Arizmendi, and H. A. Larrondo, "Estocasticidad de un atractorcaticodeterministaimplementado en fpga," Proceed- ingsIberchip 2010, Febrero 2010.
[5] J. C. Butcher, "Numerical Methods for Ordinary Differential Equations", 2nd ed., L. John Wiley & Sons, Ed., 2008.
[6] L. De Micco, H. A. Larrondo, A. Plastino, and O. A. Rosso, "Quantifiers for randomness of chaotic pseudo random number generators," Philosophical Transactions of the Royal Society A, vol. 367, pp. 3281–3296, 2009
[7] O. A. Rosso, L. DeMicco, H. A. Larrondo, M. T. Martın, and A. Plastino, "Generalized statistical complexity mea sure: a new tool for dynamical systems," International Journal of Bifurcation and Chaos, vol. 20, no. 3, pp. 775–785, 2010.
[8] J. L. Buchanan, P. R. Turner, "Numerical Methods and Analysis, McGraw-Hill, New York, 1992