МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Основы разработки встраиваемых систем на ПЛИС»

Тема: Решение систем ОДУ

Студентка гр. 1302 _____ Марзаева В.И.

Студент гр. 1302 _____ Новиков Г.В.

Студентка гр. 1302 _____ Романова О.В.

Преподаватель _____ Ханов А.В.

Санкт-Петербург

2024

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студенты Марзаева В.И., Новиков Г.В., Романова О.В.

Группа 1302

Тема работы: Решение систем ОДУ

Задание:

Реализовать метод численного решения (метод Эйлера) для системы ОДУ Langford.

Содержание пояснительной записки:

Аннотация, цель работы, теоретическое описание метода и системы, полученные результаты, листинг.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 04.10.2024

Дата сдачи реферата: 19.12.2024

Дата защиты реферата: 19.12.2024

Студентка гр. 1302	_____	Марзаева В.И.
Студент гр. 1302	_____	Новиков Г.В.
Студентка гр. 1302	_____	Романова О.В.
Преподаватель	_____	Ханов А.В.

АННОТАЦИЯ

Данная курсовая работа посвящена разработке и реализации численного метода решения системы дифференциальных уравнений с использованием языка описания аппаратуры Verilog. В работе реализован метод Эйлера для численного интегрирования с шагом h=0.01. Исходная система, начальные условия и значения параметров заданы в соответствии с представленными в задании данными. Кроме того, для проверки корректности численного решения результаты были сравнены с аналогичными расчетами, выполненными в программной среде Matlab. Это позволило оценить точность реализованного алгоритма.

SUMMARY

This coursework is dedicated to the development and implementation of a numerical method for solving a system of differential equations using the Verilog hardware description language. The Euler method was implemented for numerical integration with a step size of h=0.01. The initial system, initial conditions, and parameter values were specified in accordance with the data provided in the assignment. Additionally, to verify the accuracy of the numerical solution, the results were compared with similar calculations performed in the Matlab programming environment. This comparison allowed for an assessment of the accuracy of the implemented algorithm.

Цель работы

Разработать и реализовать алгоритм численного решения системы дифференциальных уравнений с использованием языка описания аппаратуры (Verilog). Провести моделирование поведения системы на уровне цифрового представления и сравнить с результатами, полученными в программной среде Matlab.

Теоретическая справка

В данной работе реализован метод Эйлера для решения системы Langford (рис. 1).

Данная система состоит из уравнений:

$$\frac{dx}{dt} = (z - b)x - dy$$

$$\frac{dy}{dt} = dx + (z - b)y$$

$$\frac{dz}{dt} = c + az - \frac{z^3}{3} - (x^2 + y^2)(1 + ez) + fzx^3$$

Параметры системы: a = 0.95, b = 0.7, c = 0.6, d = 3.5, e = 0.25, f = 0.1.

Начальные условия: $x_0 = 0.1$, $y_0 = 1.00$, $z_0 = 0.01$.

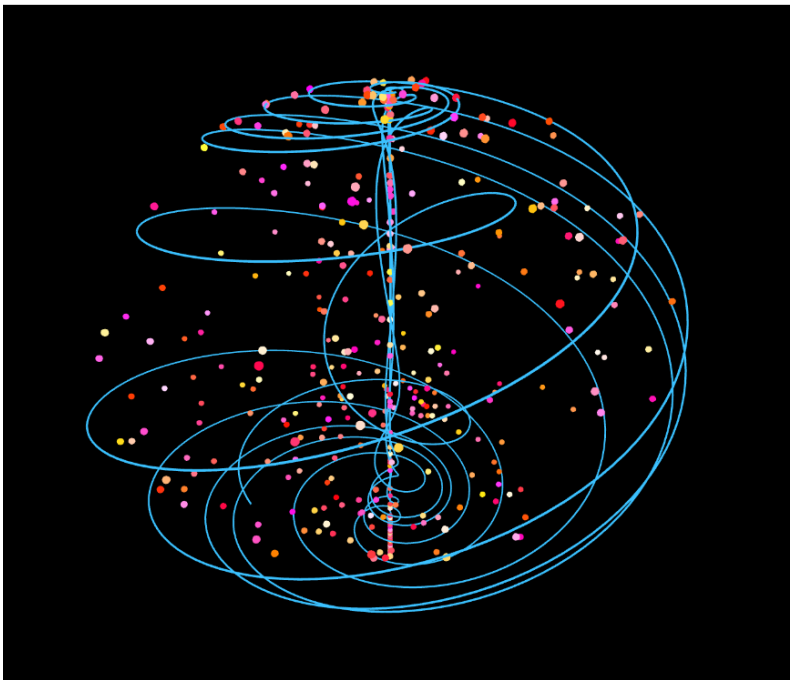

Рисунок 1 – Визуализация поведения системы Langford

Метод Эйлера является одним из простейших численных методов решения обыкновенных дифференциальных уравнений. Он основан на приближении функции прямой линией, определенной ее текущим значением и производной. Для каждого шага интегрирования h новое значение переменной рассчитывается по формуле:

$$x_{\{n+1\}} = x_n + h \cdot f(x_n, t_n)$$

В данной работе метод Эйлера используется для последовательного вычисления значений x, y и z с шагом интегрирования h=0.01. Несмотря на свою простоту, метод Эйлера требует малого шага интегрирования для достижения приемлемой точности, что было учтено при разработке модели.

Для реализации численного метода была создана модель на языке Verilog с использованием фиксированной точности (fixed-point arithmetic), что позволило учитывать аппаратные ограничения при моделировании. Результаты численного моделирования в Verilog были проверены и сравнены с расчетами, выполненными в Matlab, что позволило оценить точность реализованного алгоритма и стабильность системы.

Полученные результаты

Реализация алгоритма была выполнена в двух средах: программной (Matlab) и аппаратной (Verilog). На рис. 2-3 приведены результаты, полученные при вычислениях в среде Matlab.

```
Результаты численного решения (первые 10 шагов)
t          x1          x2          x3
0.00    0.100000    1.000000    0.010000
0.01    0.064310    0.996600    0.005970
0.02    0.028983    0.991934    0.002038
0.03   -0.005937    0.986025   -0.001795
0.04   -0.040407    0.978898   -0.005531
0.05   -0.074383    0.970577   -0.009169
0.06   -0.107826    0.961090   -0.012710
0.07   -0.140695    0.950467   -0.016154
0.08   -0.172954    0.938736   -0.019502
0.09   -0.204565    0.925928   -0.022754
Результаты сохранены в файл results.csv
```

Рисунок 2 – Результаты первых десяти вычислений

	A	B	C	D	E
979	9.78	-0.22585	-0.53419	1.8179	
980	9.79	-0.20967	-0.54807	1.8162	
981	9.8	-0.19283	-0.56152	1.8145	
982	9.81	-0.17533	-0.57453	1.8127	
983	9.82	-0.15717	-0.58706	1.8108	
984	9.83	-0.13837	-0.59908	1.8088	
985	9.84	-0.11894	-0.61057	1.8068	
986	9.85	-0.098883	-0.62149	1.8047	
987	9.86	-0.078223	-0.63181	1.8025	
988	9.87	-0.056972	-0.64152	1.8002	
989	9.88	-0.035146	-0.65057	1.7978	
990	9.89	-0.012762	-0.65894	1.7954	
991	9.9	0.010162	-0.66661	1.7929	
992	9.91	0.033604	-0.67354	1.7903	
993	9.92	0.057544	-0.6797	1.7876	
994	9.93	0.081959	-0.68508	1.7848	
995	9.94	0.10683	-0.68964	1.7819	
996	9.95	0.13212	-0.69337	1.7789	
997	9.96	0.15781	-0.69622	1.7759	
998	9.97	0.18388	-0.69819	1.7727	
999	9.98	0.21029	-0.69924	1.7695	
1000	9.99	0.23701	-0.69936	1.7661	

Рисунок 3 – Результаты последних двадцати вычислений при t = 100

На рис. 4-5 приведена визуализация системы в той же среде Matlab, построенная на рассчитанных ранее значениях. Первая диаграмма построена при t = 1000, вторая – при t = 100.

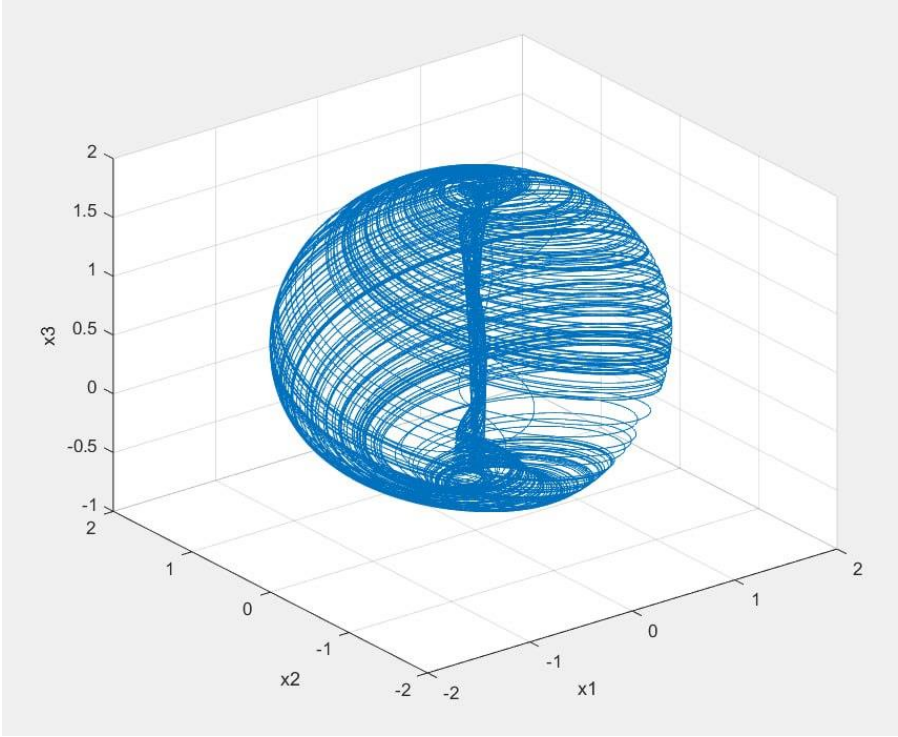

Рисунок 4 – Аттрактор Лэнгфорда при t = 1000

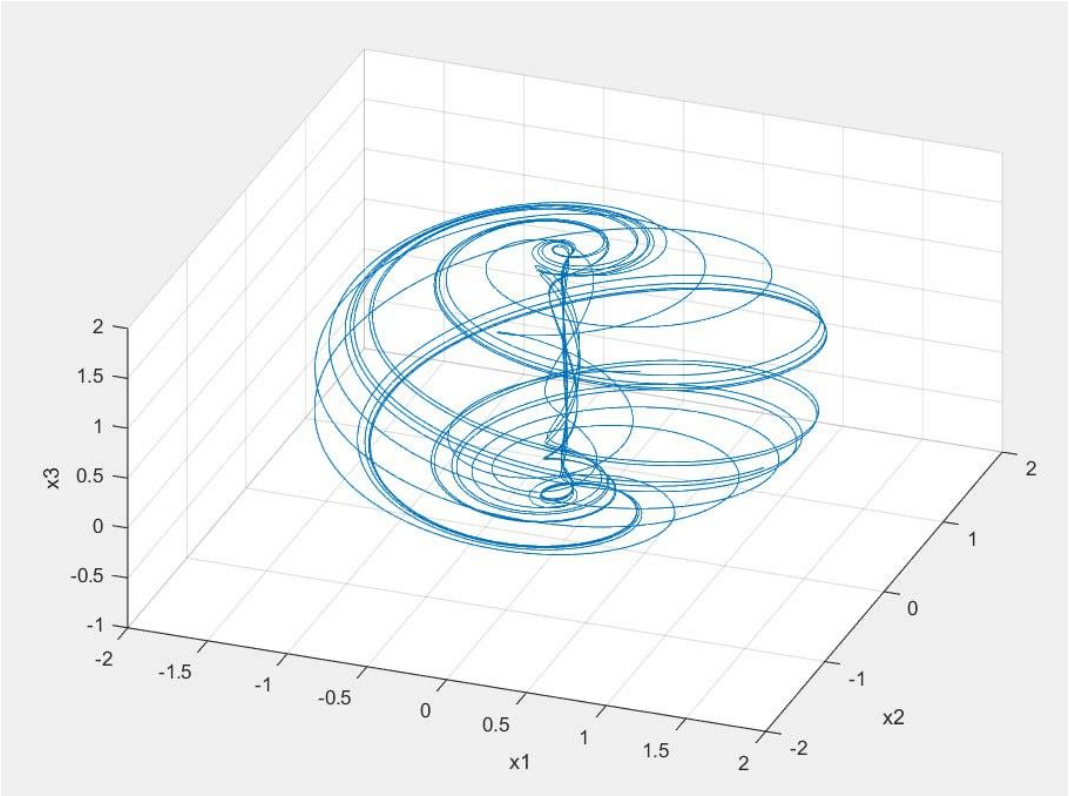

Рисунок 5 – Аттрактор Лэнгфорда при t = 100

Сравним результаты, полученные ранее, со значениями, вычисленными реализацией метода Эйлера на Verilog. На рис. 6 приведены значения, рассчитанные при первых 10 шагах, а на рис. 7 – при первых 100. Полученные значения имеют схожий результат с рассчитанными в среде Matlab.

```
PS C:\Users\faith\Desktop\kurs\please> iverilog -o ./compiled ./src/kurs.v ./src/test.v
PS C:\Users\faith\Desktop\kurs\please> vvp ./compiled
VCD info: dumpfile ./pwm_out.vcd opened for output.
Time    h_out           x_out           y_out           z_out

0       0.000000        0.100006        1.000000        0.009995

10      0.009995        0.064331        0.996597        0.005966

20      0.019989        0.029022        0.991928        0.002029

30      0.029984        -0.005890       0.986023        -0.001816

40      0.039978        -0.040344       0.978897        -0.005554

50      0.049973        -0.074310       0.970581        -0.009201

60      0.059967        -0.107742       0.961090        -0.012741

70      0.069962        -0.140610       0.950470        -0.016190

80      0.079956        -0.172852       0.938736        -0.019547

90      0.089951        -0.204453       0.925934        -0.022797
```

Рисунок 6 – Результаты вычислений при t = 10

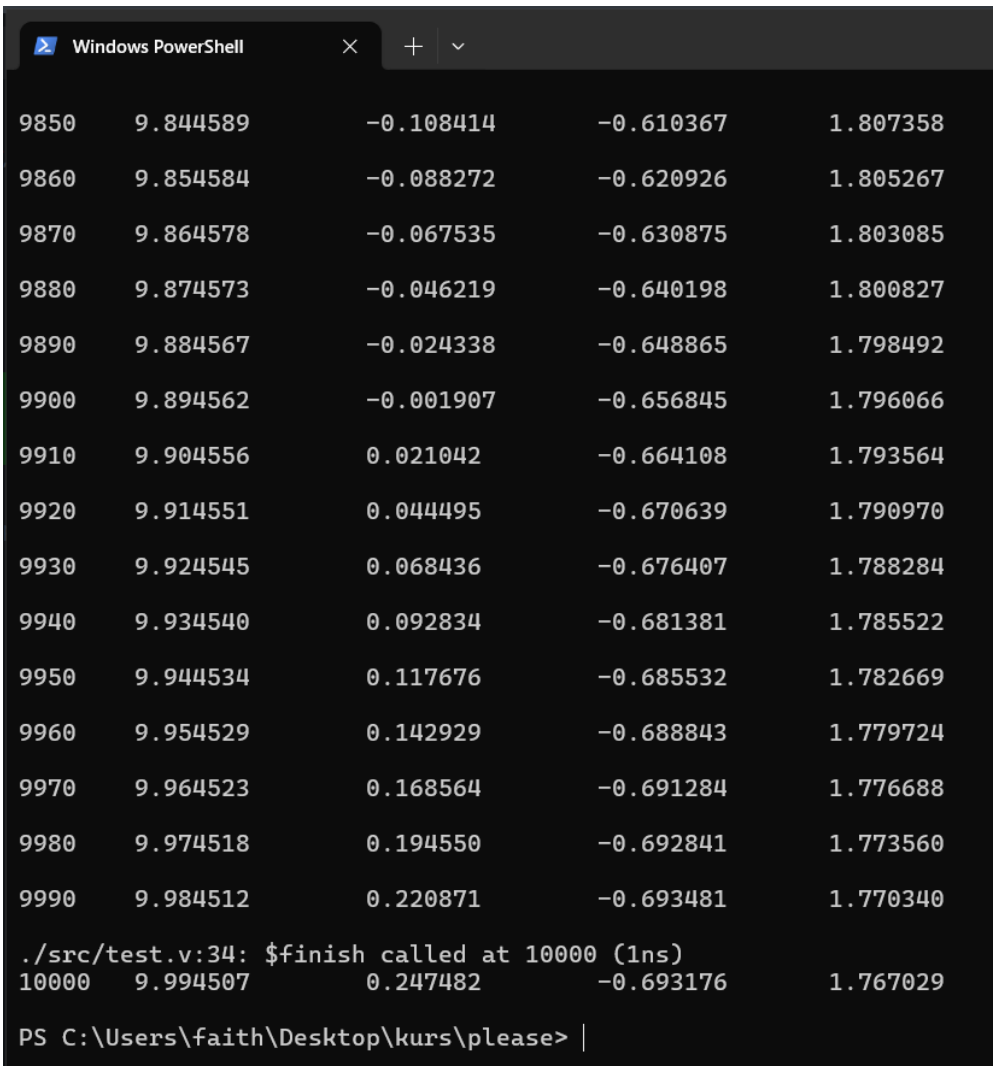

Рисунок 7 – Результаты последних вычислений при t = 100

Листинг

kurs.v

```
module pwm(
    input clk,              // Системный тактовый сигнал
    input rst,              // Сигнал сброса
    output reg signed [31:0] x_out,  // Значение x (fixed-point 16.16)
    output reg signed [31:0] y_out,  // Значение y
    output reg signed [31:0] z_out,   // Значение z
    output reg signed [31:0] h_out   // Значение шага
);

    // Параметры модели в fixed-point 16.16
    parameter signed a = 32'b0000000000000000_1111001100110011;  // 0.95
    parameter signed b = 32'b0000000000000000_1011001100110011;  // 0.7
    parameter signed c = 32'b0000000000000000_1001100110011010;  // 0.6
```

```
    parameter signed d = 32'b0000000000000011_1000000000000000;  // 3.5
    parameter signed e = 32'b0000000000000000_0100000000000000;  // 0.25
    parameter signed f = 32'b0000000000000000_0001100110011010;  // 0.1
    parameter signed h = 32'b0000000000000000_0000001010001111;  // Шаг интегрирования 0.01

    parameter signed three = 32'b0000000000000000_0101010101010101; //3.0
    parameter signed one = 32'b0000000000000001_0000000000000000;   //1.0

    // Переменные состояния x, y, z
    reg signed [31:0] x, y, z, h_temp;
    reg signed [31:0] dx, dy, dz;      // Производные

    // Временные переменные для вычислений
    reg signed [63:0] temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8;

    // Начальные условия
    initial begin
        x = 32'b0000000000000000_0001100110011010;  // x = 0.1
        y = 32'b0000000000000001_0000000000000000;  // y = 1.0
        z = 32'b0000000000000000_0000001010001111;  // z = 0.01
        h_temp = 32'b0000000000000000_0000000000000000;   //0
    end

    // Основной вычислительный блок
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            // Сброс всех переменных
            x <= 32'b0000000000000000_0001100110011010;  // x = 0.1
            y <= 32'b0000000000000001_0000000000000000;  // y = 1.0
            z <= 32'b0000000000000000_0000001010001111;  // z = 0.01
            h_temp = 32'b0000000000000000_0000000000000000;   //0
        end
        else begin
            // Вычисление dx/dt
            temp1 = (z - b) * x;
            temp2 = d * y;
            dx = (temp1 - temp2) >>> 16 & 32'hFFFFFFFF;     // dx/dt = (z - b)x - dy

            // Вычисление dy/dt
            temp1 = d * x;
            temp2 = (z - b) * y;
            dy = (temp1 + temp2) >>> 16 & 32'hFFFFFFFF;     // dy/dt = dx + (z - b)y

            // Вычисление dz/dt
```

```
            temp1 = c + ((a * z) >>> 16);               //c+az
            temp2 = (((z * z) >>> 16) * z) >>> 16;
            temp3 = (temp2 * three) >>> 16;             //z^3/3
            temp4 = (x * x) >>> 16;
            temp5 = temp4 + ((y * y) >>> 16);
            temp6 = one + ((e * z) >>> 16);
            temp7 = (temp5 * temp6) >>> 16;             //(x^2+y^2)(1+ez)
            temp8 = (((f * z) >>> 16) * ((temp4 * x) >>> 16)) >>> 16;  //fzx^3
            dz = (temp1 - temp3 - temp7 + temp8) & 32'hFFFFFFFF;

            // Метод Эйлера: обновление значений x, y, z
            x <= x + ((dx * h) >>> 16);  // x = x + h * dx
            y <= y + ((dy * h) >>> 16);  // y = y + h * dy
            z <= z + ((dz * h) >>> 16);  // z = z + h * dz
            h_temp <= h_temp + h;
        end
    end

    always @(*) begin
    // Выходные значения
        x_out = x;
        y_out = y;
        z_out = z;
        h_out = h_temp;
    end

endmodule


test.v
`timescale 1ns / 1ns

module pwm_tb();

reg clk;
reg rst;          // Сигнал сброса
wire [31:0] x_out;  // Выходное значение x
wire [31:0] y_out;  // Выходное значение y
wire [31:0] z_out;  // Выходное значение z
wire [31:0] h_out;  // Выходное значение h (шаг)

pwm r(.clk(clk), .rst(rst), .x_out(x_out), .y_out(y_out), .z_out(z_out), .h_out(h_out));

initial
begin
```

```
  clk = 0;
  rst = 1;  // Сначала активируем сброс
      #0 rst = 0;  // Снятие сброса через 10 единиц времени

      // Запуск симуляции и вывод значений
      $display("Time\th_out\t\tx_out\t\ty_out\t\tz_out\n");
      $display("%0t\t%f\t%f\t%f\t%f\n", $time, h_out / 65536.0, $signed(x_out) / 65536.0,
$signed(y_out) / 65536.0, $signed(z_out) / 65536.0);

      // Симуляция в течение 1000 тактов
      repeat(1000) begin
         #10;  // Пауза в 10 единичных времени (один такт)
         $display("%0t\t%f\t%f\t%f\t%f\n", $time, h_out / 65536.0, $signed(x_out) / 65536.0,
$signed(y_out) / 65536.0, $signed(z_out) / 65536.0);
      end

end

initial forever #(5) clk = !clk;

initial #1000 $finish;

initial
begin
  $dumpfile("./pwm_out.vcd");
  $dumpvars(0, pwm_tb);
end

endmodule
```