|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных Технологий

Кафедра Вычислительной Техники (ВТ)

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1**

«Проектирование вычислительных устройств для синтеза сигналов трансцендентных функций»

по дисциплине

«Схемотехника устройств компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент группы  ИВБО-20-23 | Деревянных В.С. |
| Принял старший преподаватель кафедры ВТ | Дуксин Н.А. |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2025 г. |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2025 г. |

Москва 2025

Аннотация

Работа включает в себя 1 рисунок, 4 листинга. Количество страниц в работе — 15.

Содержание

[Введение 4](#_Toc209229887)

[1 Постановка задачи 5](#_Toc209229888)

[2 Проектирование и реализация 6](#_Toc209229889)

[2.1 Методы решения 6](#_Toc209229890)

[2.2 Реализация 6](#_Toc209229891)

[3 ТЕСТИРОВАНИЕ 10](#_Toc209229892)

[3.1 Тестовый модуль 10](#_Toc209229893)

[Заключение 14](#_Toc209229894)

[Список использованных источников 15](#_Toc209229895)

Введение

Трансцендентные функции, такие как экспоненциальные, логарифмические и тригонометрические, являются математическим фундаментом для широкого спектра вычислительных задач. Они критически важны в областях цифровой обработки сигналов, беспроводной связи (например, для синтезаторов прямого цифрового синтеза — DDS), машинного обучения, финансового моделирования и компьютерной графики. Высокоскоростное и эффективное вычисление этих функций непосредственно на аппаратном уровне является ключевым требованием для современных высокопроизводительных систем, где программные реализации на процессорах общего назначения часто не удовлетворяют требованиям по скорости и энергопотреблению.

Трансцендентными называются функции, которые не могут быть выражены через конечное число алгебраических операций (сложения, умножения, деления и извлечения корня). Их вычисление не может быть точно представлено конечным полиномом, что требует применения специализированных аппроксимационных алгоритмов.

Прямая точная реализация трансцендентных функций на ПЛИС невозможна из-за их природы. Вместо этого используются алгоритмы, которые находят приближенное значение функции с заданной точностью. Ключевыми задачами при этом являются:

Выбор алгоритма аппроксимации: наиболее распространенные методы включая полиномиальные приближения (ряды Тейлора, Чебышёва), таблично-алгоритмические методы (CORDIC) и итерационные алгоритмы.

Оптимизация по площади и быстродействию: существует компромисс между точностью результата, количеством потребляемых аппаратных ресурсов и задержкой вычисления.

Корректная обработка чисел: Выбор формата представления данных (целочисленный фиксированной точки или число с плавающей точкой) и обеспечение контроля над ошибками округления.

1. Постановка задачи

В ходе выполнения восьмой практической работы необходимо выполнить следующие задачи:

* Изучить различные методы вычисления тригонометрических функций
* Реализовать вычисление синуса и косинуса с помощью алгоритма CORDIC
* Реализовать вычисление синуса и косинуса методом разложения в ряд Тейлора
* Реализовать вычисление синуса и косинуса с помощью таблицы готовых значений
* Провести верификацию правильности работы всех методов
* Сравнить точность предложенных методов
* Составить отчёт

1. Проектирование и реализация

## Методы решения

Всего для реализации предложено три метода:

1. Алгоритм CORDIC. Алгоритм использует последовательные повороты вектора на заранее вычисленные и зашитые в ROM углы. Для каждой итерации требуются только операции сложения/вычитания и сдвиг.
2. Аппроксимация рядом Тейлора. Программная реализация вычисляет тригонометрические функции путём их разложения в ряд Тейлора.
3. Табличный метод. Значения синуса и косинуса для всех возможных углов заранее вычислены с требуемой точностью и сохранены в памяти. Входной сигнал является адресом этой памяти, а на выходе сразу появляется соответствующее значение функции.

## Реализация

Реализация алгоритма CORDIC представлена в листинге 2.1

Листинг 2.1 — cordic.v

|  |
| --- |
| `timescale 1ns / 1ps  module cordic(  input clk, [31:0] angle, [15:0] Yin, Xin,  output [16:0] sin\_out, cos\_out  );    wire signed [31:0] atan\_table [0:30];  `include "atan\_table.vh"  reg signed [31:0] X [0:31];  reg signed [31:0] Y [0:31];  reg signed [31:0] RES\_ACC [0:31];  wire [1:0] quadrant = angle[31:30];  always@(posedge clk)  begin  case(quadrant)  2'b00, 2'b11:  begin  RES\_ACC[0] <= angle;  X[0] <= Xin;  Y[0] <= Yin;  end    2'b01:  begin  RES\_ACC[0] <= {2'b00, angle[29:0]};  X[0] <= -Yin;  Y[0] <= Xin;  end    2'b10:  begin  RES\_ACC[0] <= {2'b11, angle[29:0]};  X[0] <= Yin;  Y[0] <= -Xin;  end  endcase  end  genvar i;  generate  for (i = 0; i < 31;i = i + 1)  begin  wire rotation\_sing = RES\_ACC[i][31];  wire signed [16:0] X\_shift = X[i] >>> i;  wire signed [16:0] Y\_shift = Y[i] >>> i;    always@(posedge clk)  begin  X[i+1] <= rotation\_sing ? X[i] + Y\_shift : X[i] - Y\_shift;  Y[i+1] <= rotation\_sing ? Y[i] - X\_shift : Y[i] + X\_shift;  RES\_ACC[i+1] <= rotation\_sing ? RES\_ACC[i] + atan\_table[i] : RES\_ACC[i] - atan\_table[i];  end  end  endgenerate  assign cos\_out = X[31];  assign sin\_out = Y[31];  endmodule |

Реализация вычисления путём разложения в ряд Тейлора представлена в листинге 2.2

Листинг 2.2 — tailor.v

|  |
| --- |
| `timescale 1ns / 1ps  function automatic real real\_sin;  input real x;    real sing, x\_loc, sum;    begin  sing = 1.0;  x\_loc = x;    if (x < 0)  begin  x\_loc = -x;  sing = -1.0;  end    while (x\_loc > 3.141592652 / 2.0)  begin  x\_loc = x\_loc - 3.141592652;  sing = -1.0 \* sing;  end    sum = x\_loc - (x\_loc \*\* 3)/6 + (x\_loc\*\*5)/120 - (x\_loc\*\*7)/5040 + (x\_loc\*\*9)/362880 - (x\_loc\*\*11)/39916800;  real\_sin = sing \* sum;  end  endfunction  function automatic real real\_cos;  input real x;    real cosg, x\_loc, sum;    begin  cosg = 1.0;  x\_loc = x;    if (x < 0)  begin  x\_loc = -x;  end    while (x\_loc > 3.141592652)  begin  x\_loc = x\_loc - 3.141592652;  cosg = -1.0 \* cosg;  end    sum = 1 - (x\_loc \*\* 2)/2 + (x\_loc\*\*4)/24 - (x\_loc\*\*6)/720 + (x\_loc\*\*8)/40320 - (x\_loc\*\*10)/3628880;  real\_cos = cosg \* sum;  end  endfunction |

Реализация табличного метода представлена в листинге 2.3

Листинг 2.3 — trig\_table.v

|  |
| --- |
| `timescale 1ns / 1ps  module trig\_table#(  ANGLE\_WIDTH = 10,  VALUE\_WIDTH = 33,  COUNT = 2\*\*ANGLE\_WIDTH  )(  input [ANGLE\_WIDTH-1:0] angle\_in,  output signed [VALUE\_WIDTH-1:0] sin\_out, cos\_out  );  wire [VALUE\_WIDTH-1:0] sin\_table [0: COUNT - 1];  `include "sin\_table.vh"  wire [VALUE\_WIDTH-1:0] cos\_table [0: COUNT - 1];  `include "cos\_table.vh"  wire sin\_cond = angle\_in >= 0 && angle\_in < COUNT/2;  wire cos\_cond = (angle\_in >= 0 && angle\_in < COUNT/4) || (angle\_in >= 3 \* COUNT/4 && angle\_in < COUNT);  assign sin\_out = sin\_cond ? sin\_table[angle\_in] : -sin\_table[angle\_in];  assign cos\_out = cos\_cond ? cos\_table[angle\_in] : -cos\_table[angle\_in];  endmodule |

1. ТЕСТИРОВАНИЕ

## Тестовый модуль

Для верификации методов создадим тестовый модуль. Он представлен в листинге 3.1.

Листинг 2.3 — COMPARE\_TEST\_TRIG.v

|  |
| --- |
| `timescale 1ns / 1ps  module COMPARE\_TEST\_TRIG;  localparam pi = 3.14159265;  // Несинтезируемые функции расчётов синуса и косинуса по ряду Тейлора  // ------------------------------------------------------------------  `include "tailor.v"  reg [63:0] real\_sin\_out, real\_sin\_res\_bits;  wire [10:0] real\_sin\_exp = real\_sin\_res\_bits[62-:11];  real real\_sin\_res, real\_sin\_i;  initial  begin  real\_sin\_i = 0;  while(1)  begin  real\_sin\_res = real\_sin(real\_sin\_i);  real\_sin\_res\_bits = $realtobits(real\_sin\_res);  real\_sin\_out = $rtoi(real\_sin(real\_sin\_i) \* 2.0\*\*$signed(real\_sin\_exp - 1023 + 15));  #10;  real\_sin\_i = real\_sin\_i + pi/180;  end  end  reg [63:0] real\_cos\_out, real\_cos\_res\_bits;  wire [10:0] real\_cos\_exp = real\_cos\_res\_bits[62-:11];  real real\_cos\_res, real\_cos\_i;  initial  begin  real\_cos\_i = 0;  while(1)  begin  real\_cos\_res = real\_cos(real\_cos\_i);  real\_cos\_res\_bits = $realtobits(real\_cos\_res);  real\_cos\_out = $rtoi(real\_cos(real\_cos\_i) \* 2.0\*\*$signed(real\_cos\_exp - 1023 + 15));  #10;  real\_cos\_i = real\_cos\_i + pi/180;  end  end  // Генерация последовательности углов  // ----------------------------------  reg [63:0] i;  initial i = 0;  reg [31:0] cordic\_angle;  reg [9:0] trig\_table\_angle;  always  begin  //2^32 \* a / 360 =  trig\_table\_angle = ((1 << 10)\*i)/360;  cordic\_angle = ((1 << 32)\*i)/360;  #10;  i = i + 1;  end  // Синхросигнал  reg clk;  initial clk = 0;  always #5 clk <= ~clk;    // Модуль CORDIC  // -------------------  reg [15:0] Xin, Yin;  wire [16:0] Xout, Yout, cos\_cordic, sin\_cordic;  initial  begin  // Xin = 32000/1.647;  Xin = 32000;  Yin = 0;  end  cordic uut1 (  .clk(clk),  .angle(cordic\_angle),  .Xin(Xin),  .Yin(Yin),  .cos\_out(Xout),  .sin\_out(Yout)  );  assign cos\_cordic = Xout;  assign sin\_cordic = Yout;  // Табличный модуль  // -------------------  localparam TABLE\_VALUE\_WIDTH = 33;  localparam TABLE\_ANGLE\_WIDTH = 10;  wire [TABLE\_VALUE\_WIDTH-1:0] trig\_table\_sin;  wire [TABLE\_VALUE\_WIDTH-1:0] trig\_table\_cos;  trig\_table #(  .VALUE\_WIDTH(TABLE\_VALUE\_WIDTH),  .ANGLE\_WIDTH(TABLE\_ANGLE\_WIDTH)  ) uut  (  .angle\_in(trig\_table\_angle),  .sin\_out(trig\_table\_sin),  .cos\_out(trig\_table\_cos)  );  // IP CORDIC  wire ip\_valid\_out;  wire [31:0] sin\_ip\_out, cos\_ip\_out;  reg [31:0] ip\_cordic\_angle;  real r\_ip\_cordic\_angle;  initial  begin  r\_ip\_cordic\_angle = 0;  ip\_cordic\_angle = 0;    @(posedge clk)    while(1)  begin  @(posedge clk)  r\_ip\_cordic\_angle = r\_ip\_cordic\_angle + pi/180;  if (r\_ip\_cordic\_angle > pi)  r\_ip\_cordic\_angle = -pi;  ip\_cordic\_angle = $rtoi(r\_ip\_cordic\_angle \* (2.0\*\*29));    end    end  cordic\_0 u\_ip (  .s\_axis\_phase\_tdata(ip\_cordic\_angle),  .s\_axis\_phase\_tvalid(1'b1),  .aclk(clk),  .m\_axis\_dout\_tdata({sin\_ip\_out, cos\_ip\_out}),  .m\_axis\_dout\_tvalid(ip\_valid\_out)  );  endmodule |

Результаты тестирования представлены на рисунке 3.1.

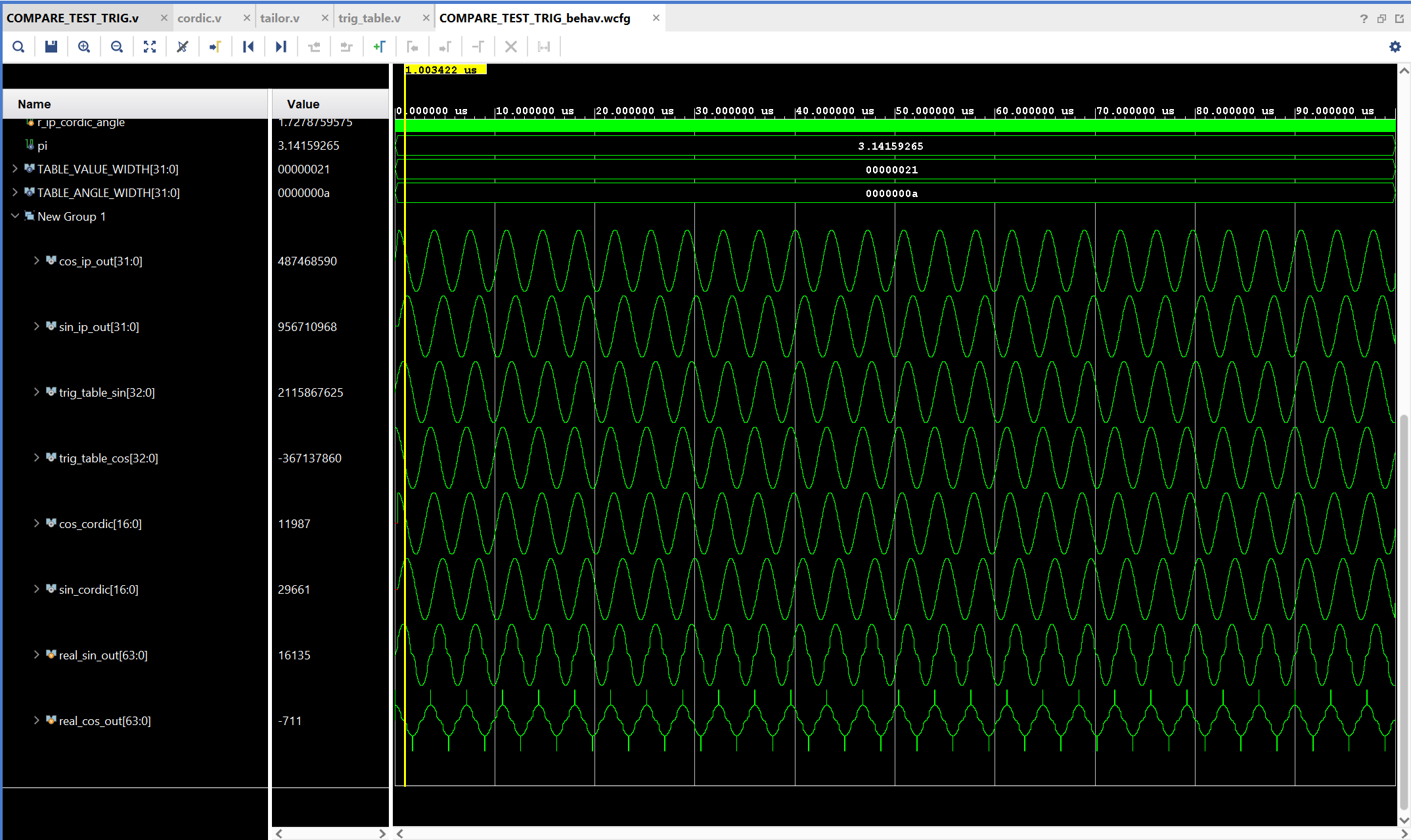


Рисунок 3.1 — Результаты тестирования

По итогам тестирования модуль CORDIC и табличный метод продемонстрировали точность, сопоставимую с IP-ядром XILINX, тогда как вычисления по рядам Тейлора оказались менее точными. Это было предсказуемо, поскольку расчёт выполнялся лишь до шестого члена ряда.

Заключение

В рамках практической работы были разработаны и проверены на практике различные способы вычисления трансцендентных функций (sin и cos) на языке Verilog. Проведён сравнительный анализ методов — ряды Тейлора, табличный способ и алгоритм CORDIC — с результатами работы готового IP-ядра XILINX.

Список использованных источников

1. Xilinx (AMD Official Website) [Электронный ресурс]. - URL: <https://www.xilinx.com/>.

2. Угрюмов Е. П. Цифровая схемотехника: учеб. пособие для вузов. —3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 816 с.: ил.

3. Тарасов И. Е. ПЛИС Xilinx/AMD. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. - М.: Горячая линия - Телеком, 2022. - 538 с.: ил.

4. Схемотехника устройств компьютерных систем. Ч. 1 [Электронный ресурс]: лабораторный практикум // Н. А. Дуксин, Д. В. Люлява, И. Е. Тарасов. — М.: РТУ МИРЭА, 2023. (дата обращения: 23.02.2025).

5. IEEE Standard Verilog® Hardware Description Language (IEEE Std 1364-2001).

6. Соловьев В. В. Основы языка проектирования цифровой аппаратуры Verilog. — М.: Горячая линия — Телеком, 2014. — 208 с.

7. Дуксин Н.А., Люлява Д.В., Долидзе И.И. Вопросы проектирования специализированных вычислителей конвейерного типа. Сборник трудов XII Международной научной конференции «ИТ – СТАНДАРТ 2023», с. 181-193.