|  |
| --- |
|  |
| 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 №** **6**

«Покрытие тестами заданного конечного автомата для выполнения операции деления»

по дисциплине

«Схемотехника устройств компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент группы  ИВБО-01-22 | Воробьев Д.М. |
| Принял ассистент кафедры ВТ | Дуксин Н.А. |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2024 г. |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2024 г. |

Москва 2024

СОДЕРЖАНИЕ

[1 ОСНОВНОЙ РАЗДЕЛ 3](#_Toc183356377)

[1.1 Конечный автомат для операции деления 3](#_Toc183356378)

[1.2 Тестирование конечного автомата для операции деления 3](#_Toc183356379)

1 ОСНОВНОЙ РАЗДЕЛ

* 1. Конечный автомат для операции деления

Опишем модуль, который реализует конечный автомат для деления двух 4-битных чисел с обработкой ошибок, например, деления на ноль.

Состояние S0 – начальное состояние, в котором обнуляются регистры и выходы. Модуль готовится к приему данных.

Состояние S1 – в нём принимается первый операнд (делимое) и сохраняется в регистр a\_reg. Переход к следующему состоянию осуществляется, если входной сигнал valid\_in активен.

Состояние S2 – принимается второй операнд (делитель) и сохраняется в регистр b\_reg. Также, если сигнал valid\_in активен, происходит переход к состоянию S3.

Состояние S3 – выполняется операция деления. Если делитель равен нулю, возникает ошибка деления на ноль, и устанавливается сигнал ошибки error\_out. Если делимое равно нулю, результат также будет равен нулю. В противном случае выполняется деление делимого на делитель и результат записывается в d\_out (Листинг 1.1).

Листинг 1.1 – Конечный автомат для операции деления на языке Verilog

module fsm\_div (

input reset,

input clk,

input valid\_in,

input [3:0] d\_in,

output reg [3:0] d\_out,

output reg valid\_out,

output reg error\_out

);

// Константы ошибок

localparam NO\_ERROR = 0, DIV\_BY\_ZERO = 1;

// Регистры операндов

reg signed [3:0] a\_reg, b\_reg;

Продолжение листинга 1.1

// Состояния конечного автомата

localparam S0 = 0, S1 = 1, S2 = 2, S3 = 3;

reg [1:0] state;

initial state = S0;

always@(posedge clk)

begin

if (reset)

state <= S0;

else

case(state)

// Сброс регистров

S0: begin

a\_reg <= 0;

b\_reg <= 0;

d\_out <= 0;

error\_out <= 0;

valid\_out <= 0;

state <= 1;

end

// Ввод первого операнда (делимого)

S1: if (valid\_in)

begin

a\_reg <= d\_in;

state <= S2;

end

// Ввод второго операнда (делителя)

S2: if (valid\_in)

begin

b\_reg <= d\_in;

state <= S3;

end

// Выполнение операции деления

S3: begin

if (b\_reg == 0)

begin

error\_out <= DIV\_BY\_ZERO;

valid\_out <= 1;

end

else if (a\_reg == 0)

begin

d\_out <= 0;

valid\_out <= 1;

end

else

begin

d\_out <= a\_reg / b\_reg;

valid\_out <= 1;

end

state <= S0;

end

Продолжение листинга 1.1

endcase

end

endmodule

* 1. Тестирование конечного автомата для операции деления

Протестируем ранее реализованный конечный автомат. Создадим класс seq\_item, содержащий уникально изменяемое (тип randc) 4-битное знаковое поле d\_in\_rand для генерации случайных значений входного сигнала d\_in.

Covergroup «fsm\_cg» определяет группы покрытия сигнала d\_in, d\_out и error\_out. Позволяет отслеживать покрытие различных состояний и значений сигналов. В каждом положительном фронте тактового сигнала вызывается метод sample для собирания данных покрытия.

Далее выполним 256 итераций (значение достаточное для покрытия всеми возможными значениями) генерации случайных значений для d\_in с помощью объекта seq\_obj (Листинг 1.2)

Листинг 1.2 – Тестовый модуль для конечного автомата на языке Verilog

`timescale 1ns / 1ps

module test;

reg clk = 0;

reg reset = 0;

reg valid\_in = 0;

reg signed [3:0] d\_in = 0;

wire signed [3:0] d\_out;

wire valid\_out;

wire error\_out;

fsm\_div uut (

.reset(reset),

.clk(clk),

.valid\_in(valid\_in),

.d\_in(d\_in),

.d\_out(d\_out),

.valid\_out(valid\_out),

.error\_out(error\_out)

);

always #5 clk = ~clk;

// случайные значения

class seq\_item;

Продолжение листинга 1.2

randc bit signed [3:0] d\_in\_rand;

endclass

seq\_item seq\_obj;

covergroup fsm\_cg @(posedge clk);

cp\_d\_in : coverpoint d\_in;

cp\_d\_out : coverpoint d\_out;

cp\_error\_out : coverpoint error\_out;

endgroup

fsm\_cg cg;

always @(posedge clk) begin

cg.sample();

end

initial begin

seq\_obj = new();

cg = new();

// Сброс

reset = 1;

@(posedge clk);

reset = 0;

@(posedge clk);

// Проверка сброса всех регистров с помощью assert

//assert (uut.state == 0) else $warning("Reset failed: state != S0");

assert (uut.a\_reg == 0)

else

$warning("Reset failed: a\_reg != 0");

assert (uut.b\_reg == 0)

else

$warning("Reset failed: b\_reg != 0");

assert (d\_out == 0)

else

$warning("Reset failed: d\_out != 0");

assert (error\_out == 0)

else

$warning("Reset failed: error\_out != 0");

assert (valid\_out == 0)

else

$warning("Reset failed: valid\_out != 0");

// Проверка ошибки при делении на 0 с помощью assert

valid\_in = 1;

d\_in = 8; // a\_reg = 8

@(posedge clk);

d\_in = 0; // b\_reg = 0

@(posedge clk);

valid\_in = 0;

@(posedge clk);

assert (error\_out == 1)

else

Продолжение листинга 1.2

$warning("Division by zero error not detected");

// Генерировать случайные значения для d\_in и проверять корректность d\_out

repeat(256) begin

seq\_obj.randomize();

d\_in = seq\_obj.d\_in\_rand;

valid\_in = 1;

@(posedge clk);

valid\_in = 0;

@(posedge clk);

if (uut.b\_reg != 0) begin

assert (uut.d\_out == (uut.a\_reg / uut.b\_reg))

else

$warning("Incorrect division result: %d / %d = %d", uut.a\_reg, uut.b\_reg, uut.d\_out);

end

end

// Вывод результатов покрытия

$display("Coverage: %0.2f %%", cg.get\_inst\_coverage());

$display("Coverage cp\_d\_in: %0.2f %%", cg.cp\_d\_in.get\_inst\_coverage());

$display("Coverage cp\_d\_out: %0.2f %%", cg.cp\_d\_out.get\_inst\_coverage());

$display("Coverage cp\_error\_out: %0.2f %% \n", cg.cp\_error\_out.get\_inst\_coverage());

end

endmodule

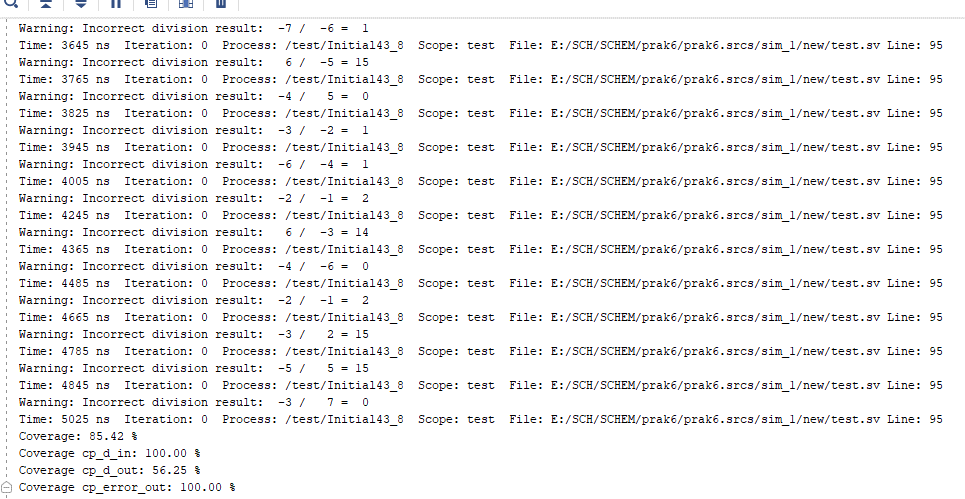
В итоге получим следующие результаты тестирования (Рисунок 1.1).

Рисунок 1.1 – Результаты тестирования

Высокий процент покрытия подтверждает, что большинство тестовых сценариев были проверены.

ЗАКЛЮЧЕНИЕ

В рамках данной практической работы реализовали покрытие необходимыми тестами заданного конечного автомата для выполнения операции деления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ПР № 1 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=405132 (Дата обращения: 23.09.2022).

2. Методические указания по ПР № 2 — URL: https://online-edu.mirea.ru/mod/resource/view.php?id=409130 (Дата обращения: 23.09.2022).

3. Смирнов С.С. Информатика [Электронный ресурс]: Методические указания по выполнению практических и лабораторных работ / С.С. Смирнов — М., МИРЭА — Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

4. Тарасов И.Е. ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. — М.: Горячая линия — Телеком, 2021. — 538 с.: ил.

5. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).

6. Антик М.И. Математическая логика и программирование в логике [Электронный ресурс]: Учебное пособие / Антик М.И., Бражникова Е.В.— М.: МИРЭА – Российский технологический университет, 2018. — 1 электрон. опт. диск (CD-ROM).

7. Жемчужникова Т.Н. Конспект лекций по дисциплине «Архитектура вычислительных машин и систем» — URL: https://drive.google.com/file/d/12OAi2\_axJ6mRr4hCbXs-mYs8Kfp4YEfj/view?us  
p=sharing (Дата обращения: 23.09.2022).

8. Антик М.И. Теория автоматов в проектировании цифровых схем [Электронный ресурс]: Учебное пособие / Антик М.И., Казанцева Л.В. — М.: МИРЭА – Российский технологический университет, 2020. — 1 электрон. опт. диск (CD-ROM).