Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №5

по дисциплине «Проектирование программного обеспечения для устройств на ПЛИС»

на тему «Знакомство с языком Verilog»

Выполнили студенты группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Беляев Д. И.

Демин М.

Приняли:

Федюнин Р.Н.

Гурин Е.И.

Пенза 2024

**Название**

Знакомство с языком Verilog.

**Цель работы**

Изучение принципов разработки схем на языке Verilog

**Ход работы**

1.1. Для заданных в табл. 1 функций на языке Verilog составить комбинационные схемы, проверить правильность их функционирования с помощью функционального моделирования (Behavioral Simulation).

Выполнить синтез схемы, просмотреть и включить в отчет результаты синтеза.

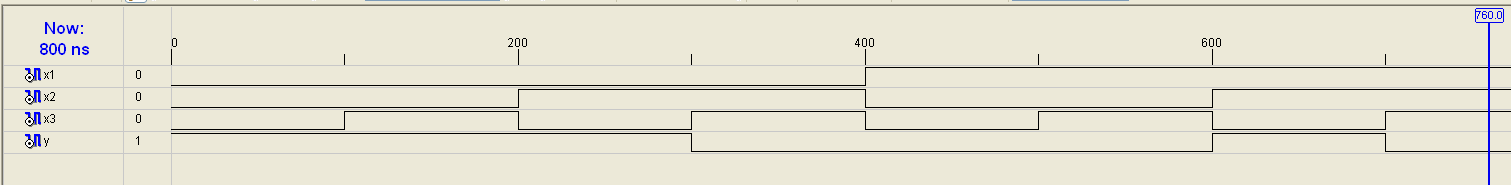


Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Вари­ант | Функция | Вари­ант | Функция |
| 1, 11 |  | 6 |  |
| 2, 12 |  | 7 |  |
| 3 |  | 8 |  |
| 4 |  | 9 |  |
| 5 |  | 10 |  |

module logic\_function(

input x1,

input x2,

input x3,

output y

);

assign y = (~x1 & ~x2) | (x1 & x2 & ~x3) | (~x1 & x2 & ~x3);

endmodule

1.2. Для заданных в табл. 1 функций на языке Verilog составить схемы c триггером на выходе. Проверить правильность их работы с помощью функционального моделирования.

Выполнить синтез схемы, просмотреть и включить в отчет результаты синтеза.

module list1 (

input clk, // Синхросигнал

input x1, // Вход x1

input x2, // Вход x2

input x3, // Вход x3

output reg y // Выход y

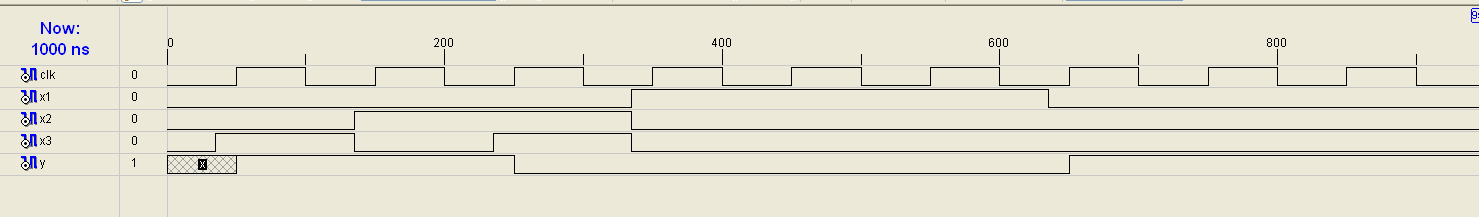
);

always @(posedge clk) begin

y <= (~x1 & ~x2) | (x1 & x2 & ~x3) | (~x1 & x2 & ~x3);

end

endmodule



2. Разработать на языке Verilog схему счетчика (разрядность N выбрать по таблице 2). Выполнить функциональное моделирование.

Таблица 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вар. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 5 | 6 | 7 | 8 | 9 |

module counter (

input clk,

input reset,

output reg [7:0] count

);

always @(posedge clk or posedge reset) begin

if (reset)

count <= 8'b0;

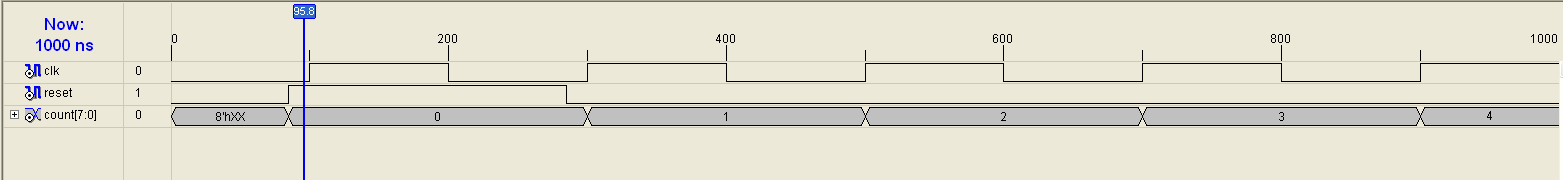
else

count <= count + 1;

end

endmodule

Выполнить синтез схемы, просмотреть и включить в отчет результаты синтеза.



3. Разработать на языке Verilog схему задержки на заданное количество тактов N.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вар. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | 5 | 4 | 3 | 2 | 5 | 6 | 4 | 3 | 2 | 4 | 3 | 2 |

module delay\_2\_ticks (

input clk,

input reset,

input in\_signal,

output reg out\_signal

);

reg [1:0] shift\_reg;

always @(posedge clk or posedge reset) begin

if (reset)

shift\_reg <= 2'b00;

else

shift\_reg <= {shift\_reg[0], in\_signal};

end

always @(posedge clk or posedge reset) begin

if (reset)

out\_signal <= 0;

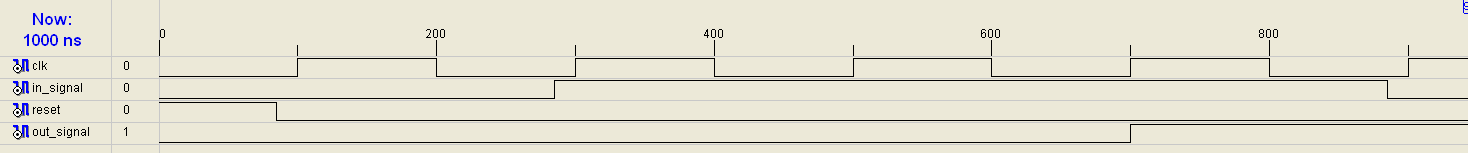
else

out\_signal <= shift\_reg[1];

end

endmodule

Выполнить функциональное моделирование. Входные сигналы сформировать с использованием тестового файла на языке Verilog



4. Разработать на языке Verilog схему формирователя.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вар. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Рис. | П1-1 | П1-2 | П1-3 | П1-4 | П1-5 | П1-6 | П1-3 | П1-4 | П1-1 | П1-2 | П1-5 | П1-6 |

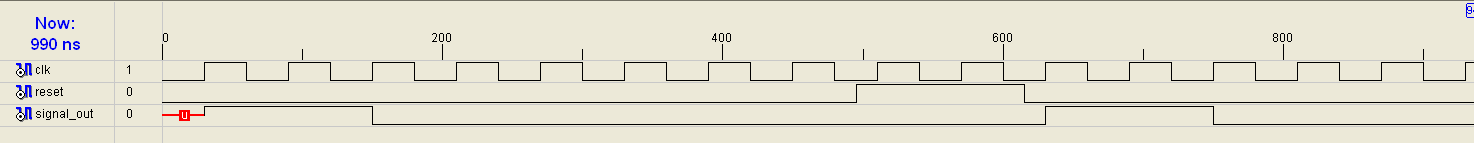
Выполнить функциональное моделирование. Входные сигналы сформировать с использованием тестового файла на языке Verilog

Рис. П1-1. Рис. П1-2.

Рис. П1-3. Рис. П1-4.

Рис. П1-5. Рис. П1-6.

module p1 (

input clk, // Синхросигнал

input reset, // Вход сброса

input in\_signal, // Входной сигнал

output reg signal\_out // Выход сигнала

);

reg [1:0] state; // Состояние (0 - не сигнальное, 1 - сигнальное, 2 - завершено)

reg clk\_prev; // Предыдущее состояние clk

always @(posedge clk or posedge reset) begin

if (reset) begin

state <= 2'b00; // Сброс состояния

signal\_out <= 1'b0; // Выход в не сигнальное состояние

clk\_prev <= 1'b0;

end else begin

if (clk && !clk\_prev) begin // Проверяем, изменился ли сигнал clk

case (state)

2'b00: begin // Если в не сигнальном состоянии

if (in\_signal) begin

state <= 2'b01; // Переход в сигнальное состояние

signal\_out <= 1'b1; // Установить выход в сигнальное состояние

end

end

2'b01: begin // Если в сигнальном состоянии

state <= 2'b10; // Переход в состояние завершения

signal\_out <= 1'b1; // Выход остается в сигнальном состоянии

end

2'b10: begin // Если в состоянии завершения

state <= 2'b00; // Переход обратно в не сигнальное состояние

signal\_out <= 1'b0; // Выход в не сигнальное состояние

end

endcase

end

clk\_prev <= clk; // Сохраняем текущее состояние clk

end

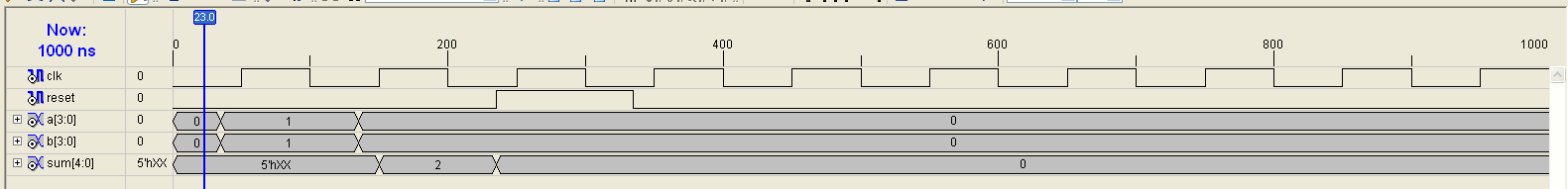
end

endmodule

5. Разработать на языке Verilog схему четырехразрядного комбинационного сумматора с регистрами на входах и выходе (рис. П2-1).



рис. П2-1



Выполнить функциональное моделирование. Входные сигналы сформировать с использованием тестового файла на языке Verilog для 5 пар слагаемых.

module adder\_4bit (

input clk,

input reset,

input [3:0] a,

input [3:0] b,

output reg [4:0] sum

);

reg [3:0] reg\_a;

reg [3:0] reg\_b;

always @(posedge clk or posedge reset) begin

if (reset) begin

reg\_a <= 4'b0;

reg\_b <= 4'b0;

sum <= 5'b0;

end else begin

reg\_a <= a;

reg\_b <= b;

sum <= reg\_a + reg\_b;

end

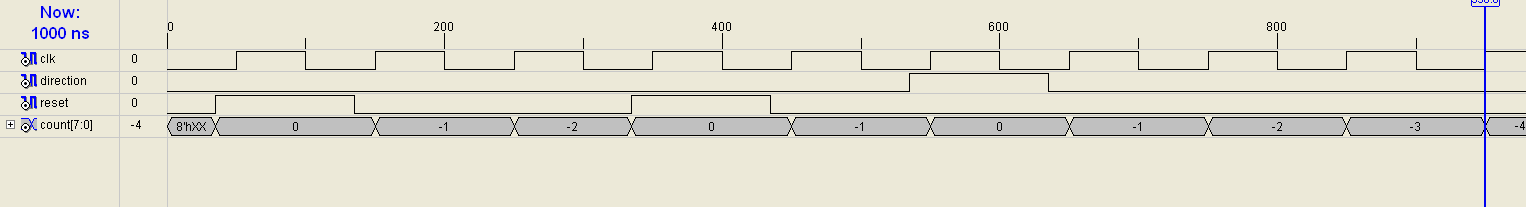
end

endmodule

6. Разработать на языке Verilog и ввести схему реверсивного счетчика (разрядность N выбрать по таблице 3). Выполнить функциональное моделирование. Выполнить реализацию проекта.

Таблица 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вар. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 11 | 10 | 9 | 8 | 7 |



module reverse\_counter (

input clk,

input reset,

input direction, // 1 для увеличения, 0 для уменьшения

output reg [7:0] count

);

always @(posedge clk or posedge reset) begin

if (reset)

count <= 8'b0;

else if (direction)

count <= count + 1;

else

count <= count - 1;

end

endmodule

**Вывод**

Изучили принципы разработки схем на языке Verilog