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 № 7

«Реализация конечных автоматов, заданных автоматным графом»

по дисциплине

«Архитектура вычислительных машин и систем»

Выполнил студент группы ИВБО-01-22 — Зырянов М.А.

Принял ассистент кафедры ВТ — Дуксина И.И.

Практическая работа выполнена «__»_____2023 г.

«Зачтено» «__»_____2023 г.

Москва 2023

АННОТАЦИЯ

Данная работа включает в себя 2 рисунка, 2 таблицы и 3 листинга. Количество страниц в работе — 14.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Требуется создать проект в САПР Vivado, создать модуль, описывающий автомат Мили согласно варианту, выданному преподавателем, произвести преобразование автомата Мили в эквивалентный ему автомат Мура и создать его модуль, создать тестовый модуль на языке Verilog HDL и произвести верификацию модулей посредством временной диаграммы. На временной диаграмме должны быть представлены все возможные переходы между состояниями, при этом каждый переход должен быть совершен хотя бы раз.

1 СОЗДАНИЕ МОДУЛЕЙ В САПР VIVADO

1.1 Вариант автомата Мили

Вариант автомата Мили представлен на Рисунке 1.1.

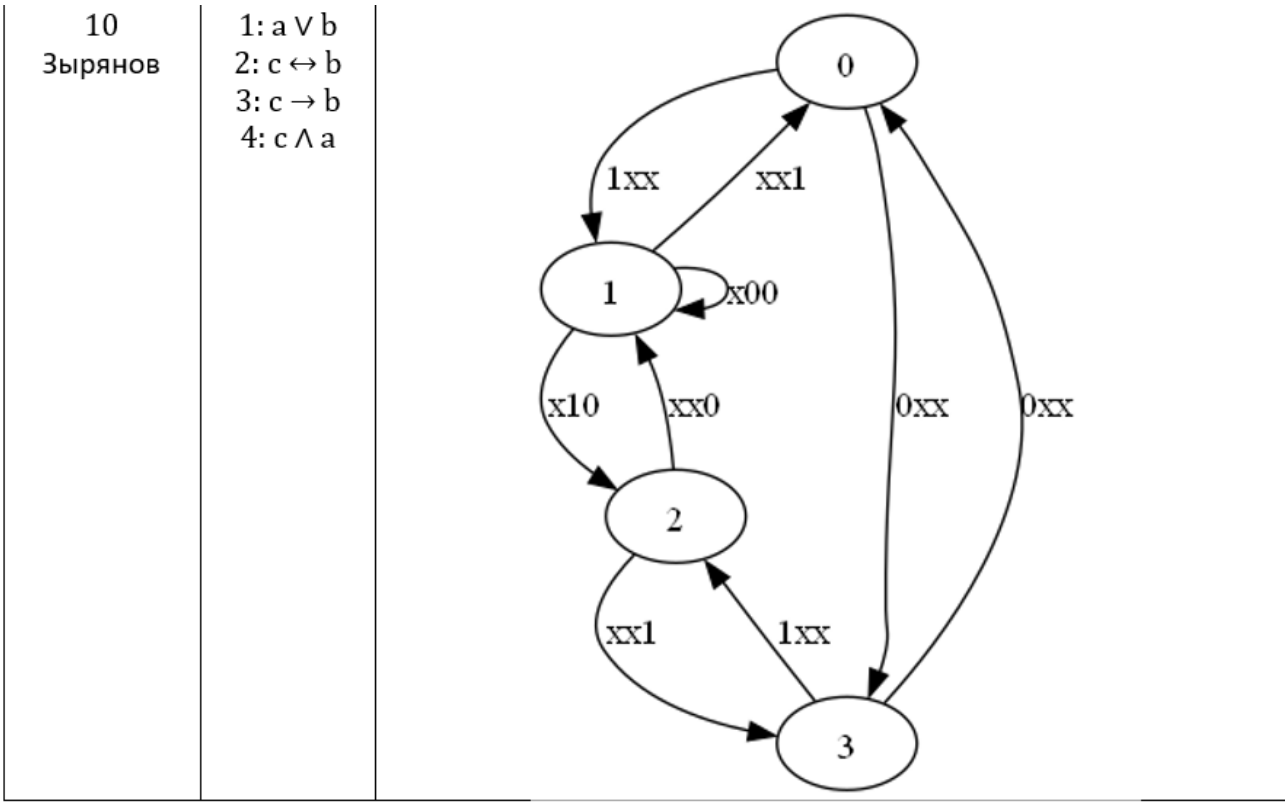

Рисунок 1.1 – Вариант автомата Мили

1.2 Описание исходного кода автомата Мили в САПР Vivado

Для создания модуля автомата Мили в САПР Vivado [4] потребуется

создать обычный модуль в директории «design sources» [1]. Модуль будет назван «Mealy_SFM».

В модуле будут четыре однобитных шины «a», «b», «c», отвечающие за входные данные, и «clk» [1], отвечающий за синхросигнал, а также

одноразрядный регистр «d», отвечающий за выходные данные. Модуль будет иметь в себе регистры «state» и «next_state», отвечающие за текущее и следующее состояния, соответственно.

На каждом переднем фронте синхросигнала регистру «state» будет присваиваться значение регистра «next_state», что эквивалентно изменению состояния нашего автомата [5]. При каждом изменении входных данных регистру «d» будет присвоено новое значение, в соответствии с условиями задачи.

Модуль автомата Мили представлен в Листинге 1.1.

Листинг 1.1 — Модуль автомата Мили

```
module Mealy_SFM(
    input a, b, c, clk,
    output reg d
    );
reg [1:0] next_state = 0;
reg [1:0] state = 0;
reg a_ = 0, b_ = 0, c_ = 0;
initial d = 1'b0;

always @(posedge clk)
    begin
        a_ = a;
        b_ = b;
        c_ = c;
    end

always @(a_, b_, c_, clk)
    begin
        casex ({state, a, b, c})
            5'b000xx: next_state <= 2'b11;
            5'b001xx: next_state <= 2'b01;
            5'b01xx1: next_state <= 2'b00;
            5'b01x00: next_state <= 2'b01;
            5'b01x10: next_state <= 2'b10;
            5'b10xx0: next_state <= 2'b01;
            5'b10xx1: next_state <= 2'b11;
            5'b111xx: next_state <= 2'b10;
            5'b110xx: next_state <= 2'b00;
            default: next_state <= next_state;
         endcase
    end

always @(a, b, c, state)
    begin
        casex (state)
            2'b00: d = a | b;
            2'b01: d = (~c & ~b) | (c & b);
            2'b10: d = ~c | b;
            2'b11: d = c & a;
            default: d = d;
        endcase
     end

always @(posedge clk)
    begin
```

```
        state = next_state;
    end
endmodule
```

1.3 Преобразование автомата Мили в эквивалентный ему автомат Мура

Составим таблицу переходов и выходов для автомата Мили (Таблица 1.1).

Таблица 1.1 – Таблица переходов и выходов автомата Мили

a,b,c/S	000	001	010	011	100	101	110	111
S0	S3/0	S3/0	S3/1	S3/1	S1/1	S1/1	S1/1	S1/1
S1	S1/1	S0/0	S2/0	S0/1	S1/1	S0/0	S2/0	S0/1
S2	S1/1	S3/0	S1/1	S3/1	S1/1	S3/0	S1/1	S3/1
S3	S0/0	S0/0	S0/0	S0/0	S2/0	S2/1	S2/0	S2/1

Преобразуем таблицу переходов и выходов автомата Мили в таблицу переходов и выходов эквивалентного автомата Мура (Таблица 1.2).

Таблица 1.2 – Таблица переходов и выходов эквивалентного автомата Мура

a,b,c/S,W	000	001	010	011	100	101	110	111
S0,0	S3,0	S3,0	S3,1	S3,1	S1,1	S1,1	S1,1	S1,1
S0,1	S3,0	S3,0	S3,1	S3,1	S1,1	S1,1	S1,1	S1,1
S1,0	S1,1	S0,0	S2,0	S0,1	S1,1	S0,0	S2,0	S0,1
S1,1	S1,1	S0,0	S2,0	S0,1	S1,1	S0,0	S2,0	S0,1
S2,0	S1,1	S3,0	S1,1	S3,1	S1,1	S3,0	S1,1	S3,1
S2,1	S1,1	S3,0	S1,1	S3,1	S1,1	S3,0	S1,1	S3,1
S3,0	S0,0	S0,0	S0,0	S0,0	S2,0	S2,1	S2,0	S2,1
S3,1	S0,0	S0,0	S0,0	S0,0	S2,0	S2,1	S2,0	S2,1

1.4 Описание исходного кода автомата Мура в САПР Vivado

Для создания модуля автомата Мура в САПР Vivado потребуется создать обычный модуль в директории «design sources». Модуль будет назван «Moore_SFM».

В модуле будут 8 параметров [2] «A», «B», «C», «D», «E», «F», «G» и «H», которые будут по умолчанию равны 0, 1, 2, 3, 4, 5, 6, 7, соответственно. Также в модуле будут четыре однобитных шины «a», «b», «c», отвечающие за входные данные, и «clk», отвечающий за синхросигнал, а также одноразрядный регистр «d», отвечающий за выходные данные. Модуль будет иметь в себе регистры

«state» и «next_state», отвечающие за текущее и следующее состояния, соответственно.

На каждом переднем фронте синхросигнала регистру «state» будет присваиваться значение регистра «next_state», что эквивалентно изменению состояния нашего автомата, и регистру «d» будет присвоено новое значение, в соответствии с состоянием автомата.

Модуль эквивалентного автомата Мура представлен в Листинге 1.2.

Листинг 1.2 — Модуль эквивалентного автомата Мура

```
module Moore_SFM #(
    parameter A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011,
    E = 3'b100, F = 3'b101, G = 3'b110, H = 3'b111)
(
    input a, b, c, clk,
    output reg d
);
reg [2:0] next_state = 3'b000;
reg [2:0] state = 3'b000;
reg a_ = 0, b_ = 0, c_ = 0;

initial d = 1'b0;

always @(posedge clk)
    begin
        a_ = a;
        b_ = b;
        c_ = c;
    end

always @(a_, b_, c_, clk)
    begin
        casex ({state, a, b, c})
            {A, 3'b000}, {B, 3'b000}: next_state <= G;
            {A, 3'b001}, {B, 3'b001}: next_state <= G;
            {A, 3'b010}, {B, 3'b010}: next_state <= H;
            {A, 3'b011}, {B, 3'b011}: next_state <= H;
            {A, 3'b100}, {B, 3'b100}: next_state <= D;
            {A, 3'b101}, {B, 3'b101}: next_state <= D;
            {A, 3'b110}, {B, 3'b110}: next_state <= D;
            {A, 3'b111}, {B, 3'b111}: next_state <= D;

            {C, 3'b000}, {D, 3'b000}: next_state <= D;
            {C, 3'b001}, {D, 3'b001}: next_state <= A;
            {C, 3'b010}, {D, 3'b010}: next_state <= E;
            {C, 3'b011}, {D, 3'b011}: next_state <= B;
            {C, 3'b100}, {D, 3'b100}: next_state <= D;
            {C, 3'b101}, {D, 3'b101}: next_state <= A;
            {C, 3'b110}, {D, 3'b110}: next_state <= E;
            {C, 3'b111}, {D, 3'b111}: next_state <= B;

            {E, 3'b000}, {F, 3'b000}: next_state <= D;
```

```
            {E, 3'b001}, {F, 3'b001}: next_state <= G;
            {E, 3'b010}, {F, 3'b010}: next_state <= D;
            {E, 3'b011}, {F, 3'b011}: next_state <= H;
            {E, 3'b100}, {F, 3'b100}: next_state <= D;
            {E, 3'b101}, {F, 3'b101}: next_state <= G;
            {E, 3'b110}, {F, 3'b110}: next_state <= D;
            {E, 3'b111}, {F, 3'b111}: next_state <= H;

            {G, 3'b000}, {H, 3'b000}: next_state <= A;
            {G, 3'b001}, {H, 3'b001}: next_state <= A;
            {G, 3'b010}, {H, 3'b010}: next_state <= A;
            {G, 3'b011}, {H, 3'b011}: next_state <= A;
            {G, 3'b100}, {H, 3'b100}: next_state <= E;
            {G, 3'b101}, {H, 3'b101}: next_state <= F;
            {G, 3'b110}, {H, 3'b110}: next_state <= E;
            {G, 3'b111}, {H, 3'b111}: next_state <= F;
        endcase
    end

always @(posedge clk)
    begin
        state = next_state;
        casex(state)
            3'bxx1: d = 1'b1;
            default: d = 1'b0;
        endcase
    end
endmodule
```

1.5 Описание исходного кода верификатора в САПР Vivado

Для создания модуля верификатора в САПР Vivado потребуется создать тестовый модуль в директории «simulation sources» [1]. Модуль будет назван «testbench».

В модуле будут созданы шестиразрядный регистр «args», отвечающий за номер итерации, четыре одноразрядных регистра «clk», отвечающий за синхросигнал, «a», «b» и «c», отвечающие за входные данные автоматов, две одноразрядных шины «res_mealy» и «res_moore», отвечающие за выходы автоматов Мили и Мура, соответственно.

Регистрам «args», «clk», «a», «b» и «c» при объявлении присвоим значение ноль. Генерироваться такты, путем инвертирования «clk» каждые 5 единиц времени, с помощью блока «always». С помощью условия «@(posedge)» настраиваем регистры «args», «a», «b» и «c» на передний фронт

синхроимпульсов. Регистрам «a», «b» и «c» присваивается значение, в соответствии со значением регистра «args».

Код модуля верификатора представлен в Листинге 1.3.

Листинг 1.3 — Модуль верификатора автоматов Мили и Мура

```
`timescale 1ns / 1ps

module testbench();

reg [5:0] args = 0;
reg clk = 0;
reg a = 0, b = 0, c = 0;
wire res_mealy;
wire res_moore;

always #5 clk = ~clk;
always @(posedge clk)
    begin
        case (args)
            6'b000000: begin a = 0; b = 0; c = 0; end
            6'b000001: begin a = 0; b = 0; c = 1; end
            6'b000010: begin a = 0; b = 0; c = 0; end
            6'b000011: begin a = 0; b = 1; c = 1; end

            6'b000100: begin a = 0; b = 0; c = 0; end
            6'b000101: begin a = 0; b = 1; c = 0; end
            6'b000110: begin a = 0; b = 0; c = 0; end
            6'b000111: begin a = 1; b = 1; c = 1; end

            6'b001000: begin a = 0; b = 0; c = 1; end
            6'b001001: begin a = 1; b = 1; c = 0; end
            6'b001010: begin a = 0; b = 0; c = 1; end
            6'b001011: begin a = 0; b = 0; c = 1; end

            6'b001100: begin a = 0; b = 0; c = 1; end
            6'b001101: begin a = 0; b = 1; c = 1; end
            6'b001110: begin a = 0; b = 1; c = 0; end
            6'b001111: begin a = 0; b = 1; c = 1; end

            6'b010000: begin a = 1; b = 0; c = 0; end
            6'b010001: begin a = 1; b = 1; c = 1; end
            6'b010010: begin a = 1; b = 1; c = 0; end
            6'b010011: begin a = 1; b = 1; c = 1; end

            6'b010100: begin a = 0; b = 1; c = 0; end
            6'b010101: begin a = 0; b = 1; c = 0; end
            6'b010110: begin a = 1; b = 0; c = 0; end
            6'b010111: begin a = 0; b = 0; c = 1; end

            6'b011000: begin a = 1; b = 1; c = 0; end
            6'b011001: begin a = 0; b = 0; c = 1; end
            6'b011010: begin a = 0; b = 1; c = 0; end
            6'b011011: begin a = 1; b = 1; c = 1; end
```

```
            6'b011100: begin a = 0; b = 0; c = 0; end
            6'b011101: begin a = 0; b = 1; c = 0; end
            6'b011110: begin a = 0; b = 1; c = 1; end
            6'b011111: begin a = 1; b = 1; c = 0; end

            6'b100000: begin a = 1; b = 1; c = 0; end
            6'b100001: begin a = 0; b = 0; c = 1; end
            6'b100010: begin a = 0; b = 0; c = 1; end
            6'b100011: begin a = 1; b = 1; c = 0; end

            6'b100100: begin a = 0; b = 0; c = 0; end
            6'b100101: begin a = 0; b = 1; c = 1; end
            6'b100110: begin a = 0; b = 0; c = 0; end
            6'b100111: begin a = 0; b = 0; c = 1; end

            6'b101000: begin a = 0; b = 1; c = 1; end
            6'b101001: begin a = 1; b = 0; c = 0; end
            6'b101010: begin a = 0; b = 0; c = 0; end
            6'b101011: begin a = 0; b = 0; c = 1; end

            6'b101100: begin a = 0; b = 1; c = 1; end
            6'b101101: begin a = 1; b = 0; c = 0; end
            6'b101110: begin a = 1; b = 1; c = 0; end
            6'b101111: begin a = 0; b = 0; c = 1; end

            6'b110000: begin a = 0; b = 1; c = 1; end
            6'b110001: begin a = 1; b = 0; c = 0; end
            6'b110010: begin a = 1; b = 1; c = 1; end
            6'b110011: begin a = 0; b = 0; c = 0; end

            default: $finish;
         endcase
         args = args + 1;
      end
Mealy_SFM uut1 (.a(a), .b(b), .c(c), .clk(clk), .d(res_mealy));
Moore_SFM uut2 (.a(a), .b(b), .c(c), .clk(clk), .d(res_moore));

endmodule
```

2 ВЕРИФИКАЦИЯ МОДУЛЕЙ

Произведем верификацию модулей посредством временных диаграмм (Рисунок 2.1).

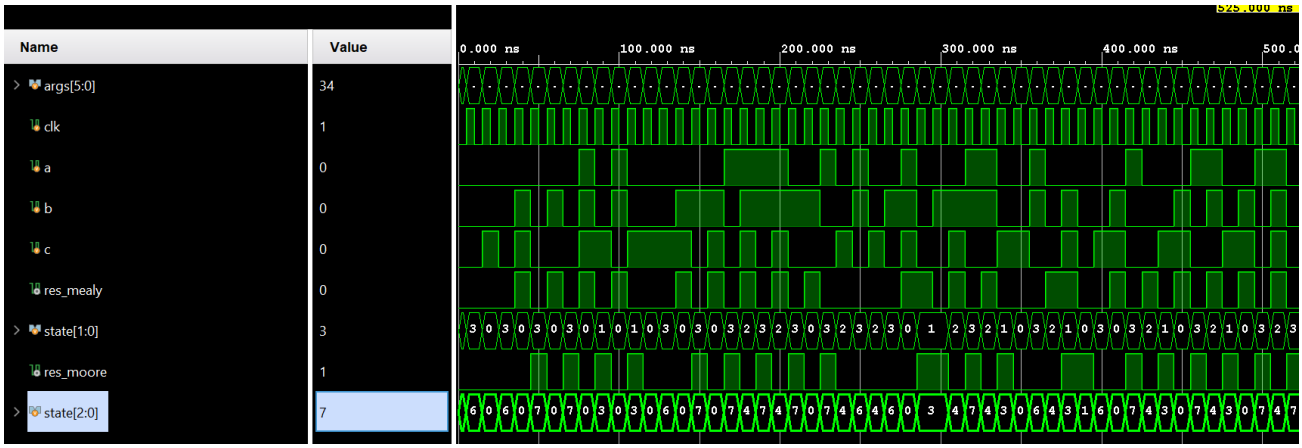

Рисунок 2.1 – Временная диаграмма

На данной работе видно, что автоматы Мили и Мура переключаются в соответствии с порядком, представленном в тестовом модуле, автомат Мура отстает от автомата Мили на 1 такт.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной практической работы были созданы модули автомата Мили и эквивалентного ему автомата Мура и проверены на корректность на языке описания аппаратуры Verilog HDL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания по ПР № 1 — URL: https://onlineedu.mirea.ru/mod/resource/view.php?id=405132 (Дата обращения: 13.09.2023).

2. Методические указания по ПР № 2 — URL: https://onlineedu.mirea.ru/mod/resource/view.php?id=409130 (Дата обращения: 13.09.2023).

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).