

Системотехника интегральных вычислителей

Задания для лабораторных работ

Весенний семестр 2010. Группа 5114.

Лабораторная работа №1

Тема работы: кодирование конечного автомата.

Цель работы: приобретение навыков реализации конечных автоматов Мура/Мили на языке Verilog HDL, изучение вопросов сопряжения реализованных блоков с другими частями проекта.

Описание работы:

В заданиях по вариантам представлен конечный автомат в форме Мура или Мили, который нужно закодировать в виде отдельного модуля на языке Verilog HDL. После этого необходимо преобразовать конечный автомат в другую форму: если в задании дан автомат Мура, то нужно преобразовать его в автомат Мили, и наоборот. Преобразованный автомат также необходимо закодировать, оформив его в виде отдельного модуля. Идентичность работы реализованных автоматов необходимо проверить с помощью testbench.

Конечный автомат имеет следующие особенности:

- большими латинскими буквами A, B, C, D, ... обозначены имена состояний автомата;
- маленькими латинскими буквами a, b, c, d, ... обозначены имена входных однобитовых сигналов, значения на которых анализируются автоматом;
- имена выходных сигналов автомата обозначаются в виде O_x, где x – прописная буква латинского алфавита (OA, OB, OC, OD, ...);
- начальное состояние автомата всегда A;
- кроме входных сигналов, непосредственно влияющих на логику работы автомата, на вход реализующего конечный автомат модуля подаются следующие сигналы:
 - Clk – сигнал синхронизации. Все переходы между состояниями, а также изменения значений выходов осуществляются *по положительному перепаду* на этом входе;
 - Reset – сигнал асинхронного сброса. Если этот сигнал в лог. «1», то конечный автомат приводится в начальное состояние. При этом, для автомата Мили значения выходных сигналов равны лог. «0». Этот сигнал может быть установлен в лог. «1» *в любой момент времени*, то есть асинхронно по отношению к Clk, и реакция модуля на него *не* должна быть задержана до следующего перепада Clk;

Тестирование работы автомата *обязательно* должно включать асинхронную (по отношению к входу Clk автомата) подачу входных сигналов.

Пример:

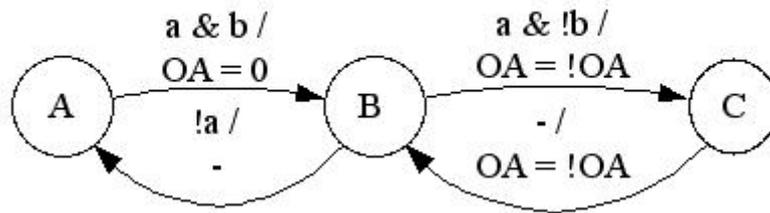


Рисунок 1. Пример конечного автомата Мили

На рисунке 1 изображен конечный автомат Мили, который имеет три состояния: А, В и С. На логику его работы влияют два сигнала: а и b. Конечный автомат имеет один выход: ОА. В момент перехода из состояния А в состояние В конечный автомат устанавливает на этом выходе лог. «0». Переход выполняется только в том случае, если истинно выражение «а & b», то есть оба входа содержат лог. «1». В момент перехода из состояния В в состояние С конечный автомат инвертирует значение на своем выходе («ОА = !ОА»). Переход из состояния С в состояние В производится всегда после того, как автомат попал в состояние С (на следующем такте конечного автомата). В момент перехода из состояния В в состояние А не производится никаких действий (кроме смены состояния).

Модуль, который реализует этот конечный автомат, имеет вход Clk. Все условия для переходов анализируются только по положительному перепаду на этом входе. Состояние конечного автомата, а также значение его выходного сигнала не изменяется до следующего положительного перепада Clk. Кроме этого входа, модуль имеет вход Reset, лог. «1» на котором «мгновенно» (не дожидаясь следующего перепада Clk) сбрасывает конечный автомат в состояние А и устанавливает на его выходе лог. «0».

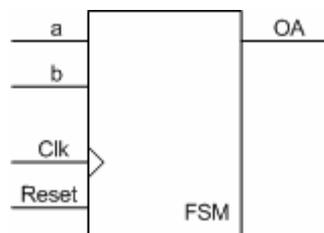


Рисунок 2. Входы и выходы конечного автомата

Исходный код реализации на Verilog выглядит следующим образом.

```

module FSM(
    input    a,
    input    b,
    input    Clk,
    input    Reset,

    output reg OA // Значение на выходе не меняется до следующего
                  // перепада Clk
);

// Символические имена состояний
parameter A = 2'b00, B = 2'b01, C = 2'b10;

// Текущее и следующее состояние
reg [1:0] state, next_state;

// Следующее значение выхода
reg next_oa;
  
```

```

// Выполнение перехода
always @(posedge Clk, posedge Reset)
    if(Reset) // Сброс имеет приоритет перед сигналом Clk
        state <= A; // Начальное состояние автомата
    else
        state <= next_state;

// Логика переходов (комбинационная схема)
always @(a, b, state, OA)
begin
    next_state = state;
    next_oa = OA;

    case(state)
        A:
            if(a & b)
            begin
                next_state = B;
                next_oa = 1'b0;
            end

        B:
            begin
                if( !a )
                begin
                    next_state = A;
                    next_oa = OA;
                end

                if(a & !b)
                begin
                    next_state = C;
                    next_oa = !OA;
                end
            end

        C:
            begin
                next_state = B;
                next_oa = !OA;
            end

        default:
            next_state = 2'bxx; // Для индикации некорректного
                               // состояния
    endcase
end

// Установка значения выходного сигнала в момент перехода
always @(posedge Clk, posedge Reset)
    if(Reset)
        OA <= 1'b0; // По Reset на выходе OA содержится лог. "0"
    else
        OA <= next_oa;

endmodule

```

Рекомендации по реализации Вашего конечного автомата:

1. Следует защелкивать все выходные сигналы в элементах памяти, если они формируются комбинационной схемой.
2. Перед кодированием рекомендуется ознакомиться с выложенной в соответствующем разделе форума

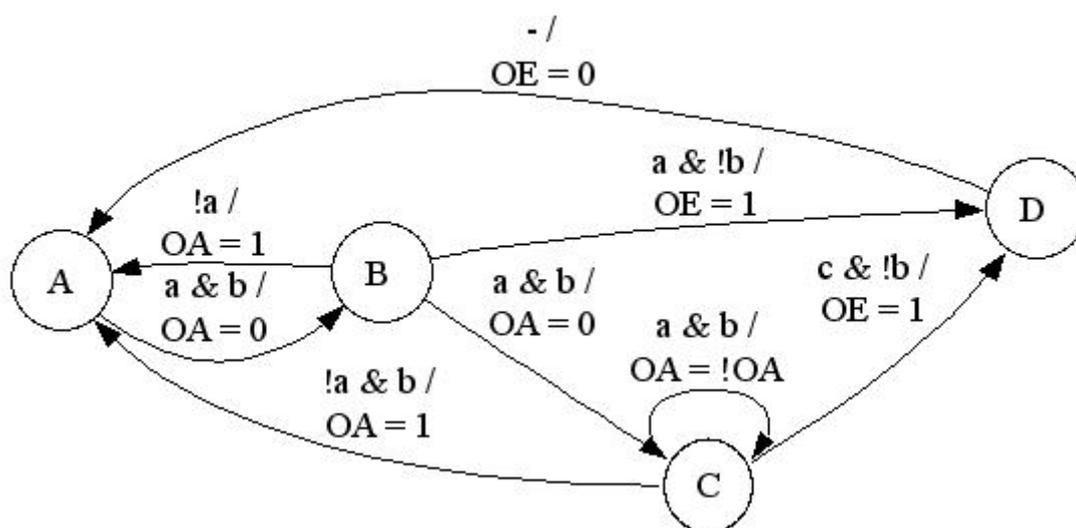
(<http://embedded.ifmo.ru/forum/viewforum.php?f=59>) литературой, касающейся вопроса кодирования конечных автоматов на Verilog.

Порядок выполнения:

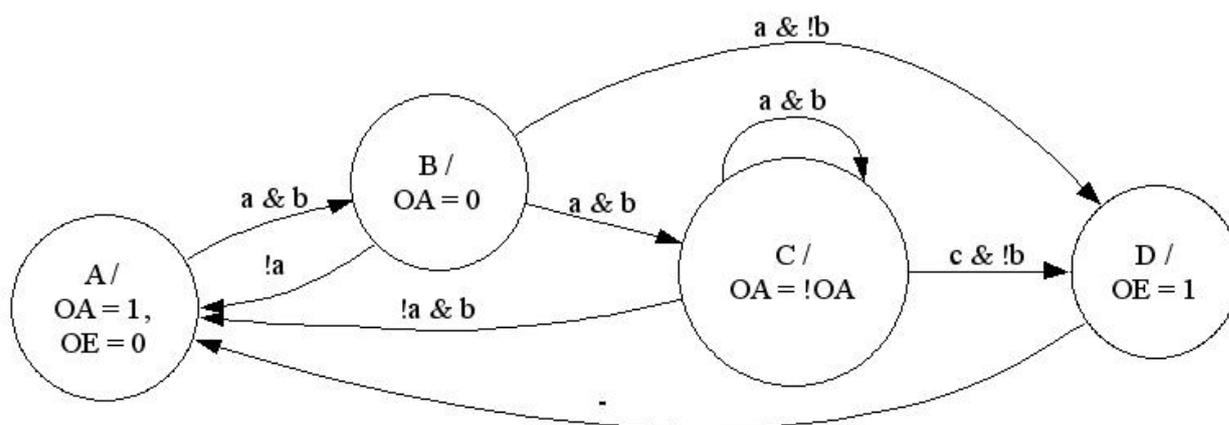
1. Закодировать на Verilog автомат, представленный в задании.
2. Проверить правильность его работы с помощью составленного testbench.
3. Преобразовать автомат в другую форму (Мура в Мили, Мили в Мура) и закодировать его.
4. Проверить правильность его работы с помощью составленного testbench.
5. Составить testbench для проверки идентичности работы двух автоматов и проверить ее.
6. Оформить отчет и защитить работу. Отчет должен включать формальное описание (лучше, в графическом виде) реализации первого и второго автоматов, схемы тестирования автоматов отдельно и тестирования идентичности работы автоматов. К отчету необходимо приложить (иметь при себе) работоспособный проект, включающий все компоненты (автоматы, testbench).

Варианты заданий:

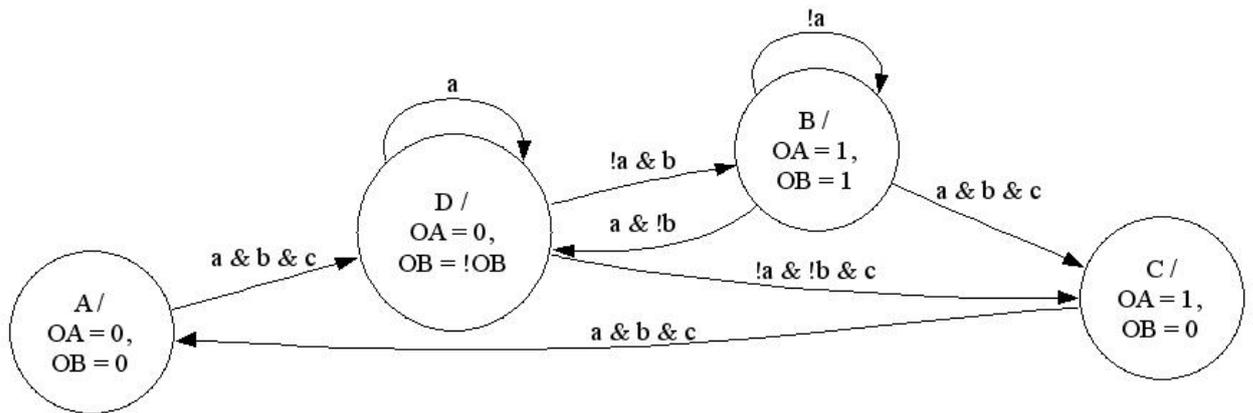
Вариант 1



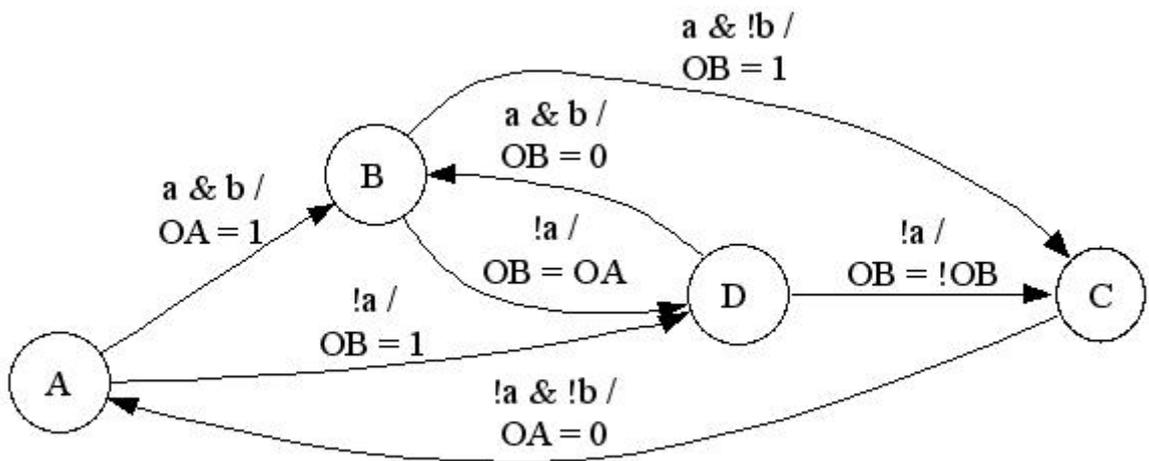
Вариант 2



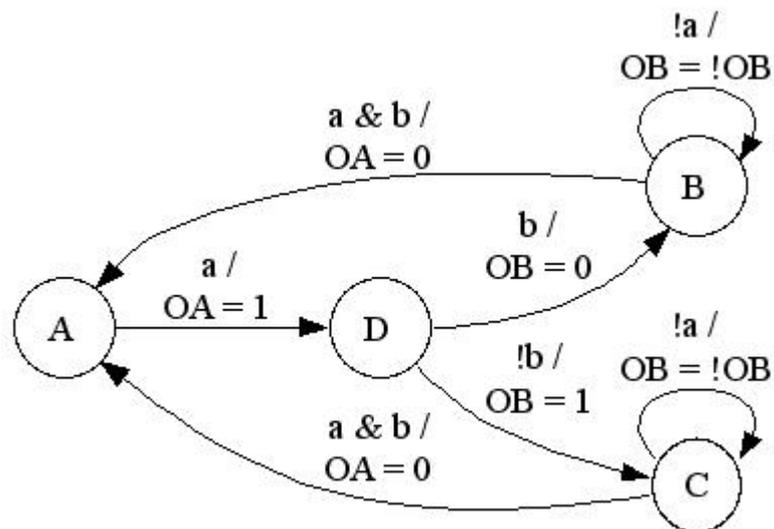
Вариант 3



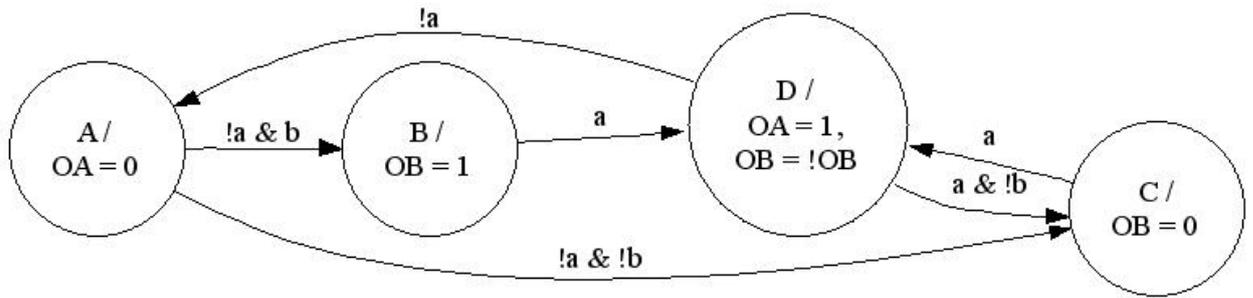
Вариант 4



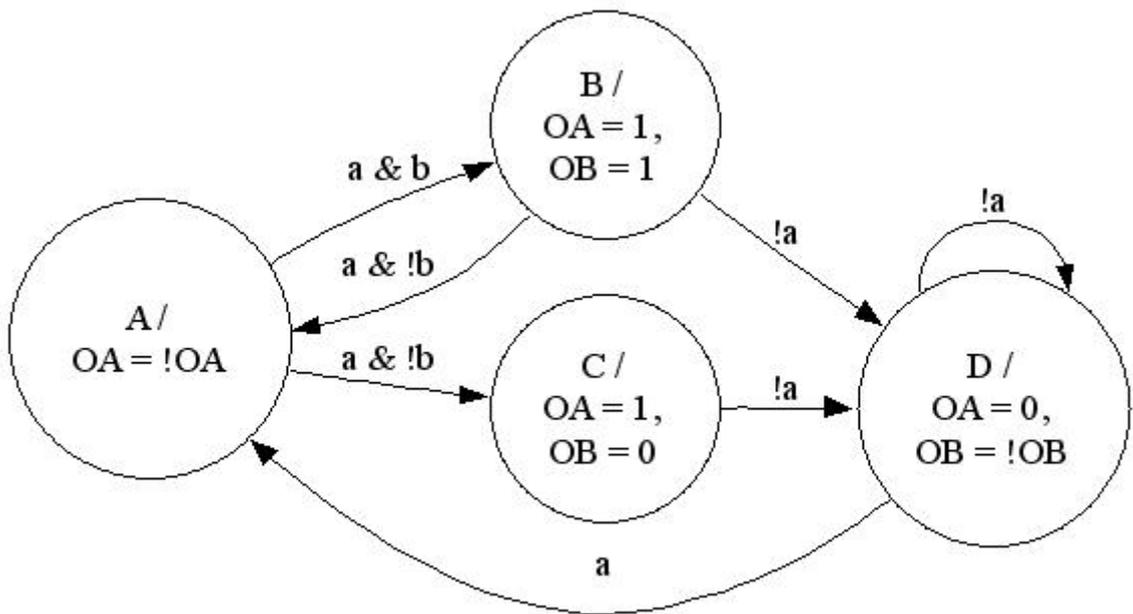
Вариант 5



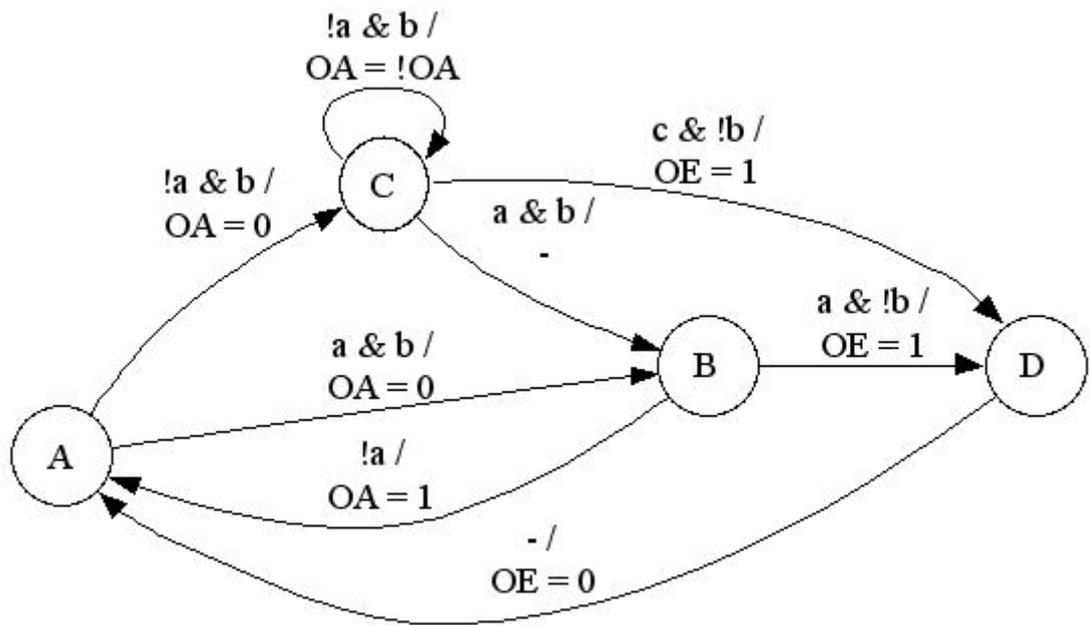
Вариант 6



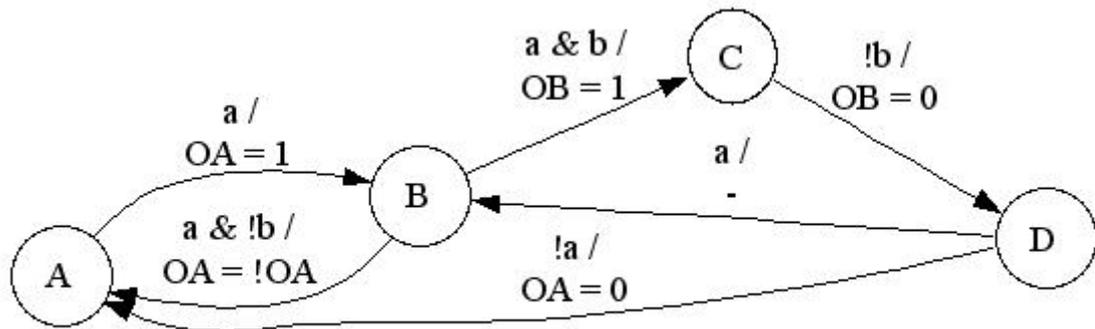
Вариант 7



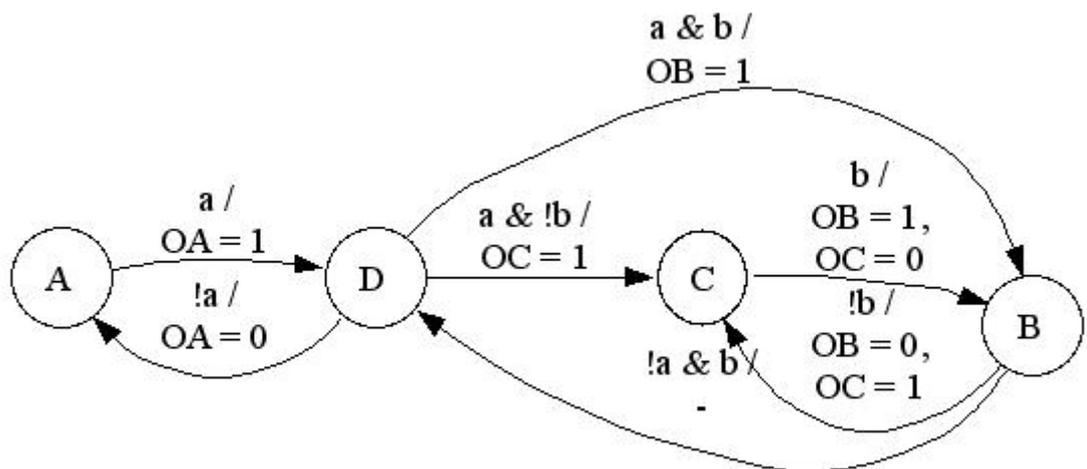
Вариант 8



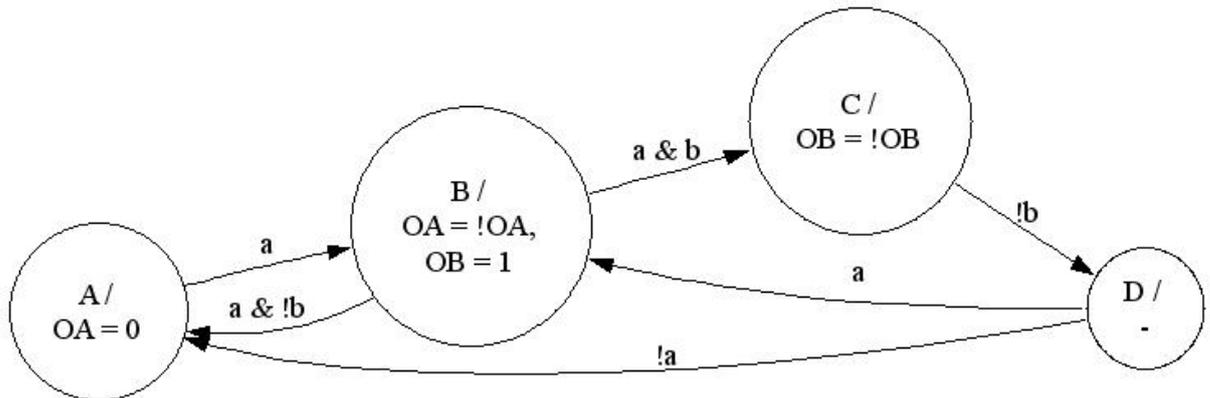
Вариант 9



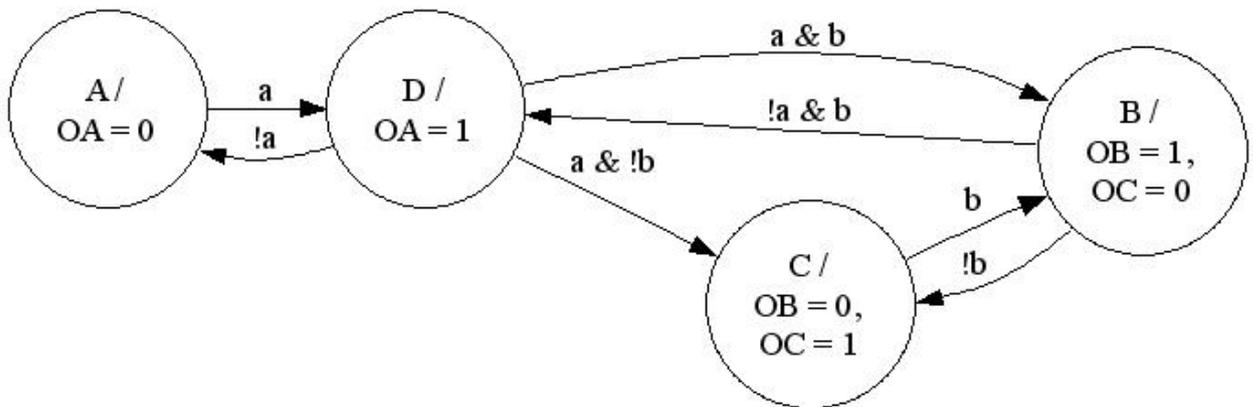
Вариант 10



Вариант 11



Вариант 12



Сроки выполнения:
10 марта 2010