ФГБОУ ВО “Воронежский государственный
технический университет”

Кафедра полупроводниковой электроники
и наноэлектроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ № 1-3
по дисциплине
«Проектирование цифровых устройств в базисе ПЛИС»
для студентов направления
11.03.04 «Электроника и наноэлектроника»
(направленность «Микроэлектроника и твердотельная электроника»)
очной формы обучения

Воронеж 2016

Составители: д-р техн. наук А.В. Строгонов, канд. техн. наук Н.Н. Кошелева, канд. техн. наук А.В. Арсентьев, ассистент А.А. Винокуров

УДК 621.382

Методические указания к выполнению лабораторных работ № 1-3 по дисциплине «Проектирование цифровых устройств в базисе ПЛИС» для студентов направления 11.03.04 «Электроника и наноэлектроника» (направленность «Микроэлектроника и твердотельная электроника») очной формы обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост. А.В. Строгонов, Н.Н. Кошелева, А.В. Арсентьев, А.А. Винокуров. Воронеж, 2016. 39 с.

В методических указаниях приведены примеры по проектированию цифровых автоматов в системе визуально-имитационного моделирования Matlab/Simulink для последующей реализации в базисе ПЛИС Xilinx.

Методические указания подготовлены в электронном виде и содержатся в файле МУ_цифр_авт_2016.pdf.

Ил. 19. Библиогр.: 5 назв.

Рецензент д-р техн. наук, проф. М.И. Горлов

Ответственный за выпуск зав. кафедрой
д-р физ.-мат. наук, проф. С.И. Рембеза

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ЛАБОРАТОРНАЯ РАБОТА № 1

Разработка цифровых автоматов с использованием пакета расширения Stateflow системы визуально-имитационного моделирования Matlab/Simulink для последующей реализации в базисе ПЛИС

Цель работы: демонстрация возможностей пакетов расширения Stateflow и Xilinx System Generator системы Matlab/Simulink по проектированию цифровых (конечных) автоматов с последующей их реализацией в базисе ПЛИС Xilinx.

Задание:

1) с использованием Stateflow-диаграммы разработайте модель конечного автомата Мили в системе Matlab/Simulink согласно выбранному варианту задания;

2) проведите имитационное моделирование цифрового автомата в системе Matlab/Simulink;

3) с помощью Simulink HDL Coder системы Matlab/Simulink извлеките VHDL-код цифрового автомата;

4) в САПР Xilinx ISE разработайте функциональную модель цифрового автомата и осуществите его моделирование;

5) сделайте выводы и оцените рабочую частоту проекта.

Теоретические сведения

Теоретические сведения по конечным автоматам можно получить в учебном пособии [1], раздел 7.3.4 “Анализ конечных автоматов с D-триггерами” и в работе [2].

Обычно рассматривают два типа автоматов – автомат Мили (Mealy) и Мура (Moore). Выход автомата Мура является функцией только текущего состояния, выход автомата Мили – функция, как текущего состояния, так и начального внешнего воздействия. Конечный автомат состоит из трех основных частей (рис. 1) [1]:

Регистр текущего состояния. Этот регистр представляет собой набор тактируемых D-триггеров, синхронизируемых

одним синхросигналом, используемый для хранения кода текущего состояния автомата. Для автомата с n состояниями требуется $Log_2(n)$ триггеров;

Логика переходов. Конечный автомат может находиться в каждый конкретный момент времени только в одном состоянии. Каждый тактовый импульс вызывает переход автомата из одного состояния в другое. Правила перехода определяются комбинационной схемой, называемой логикой переходов. Следующее состояние определяется как функция текущего состояния и входного воздействия;

Логика формирования выхода. Выход цифрового автомата обычно определяется как функция текущего состояния и исходной установки (в случае автомата Мили). Формирование выходного сигнала автомата определяется с помощью логики формирования выхода.

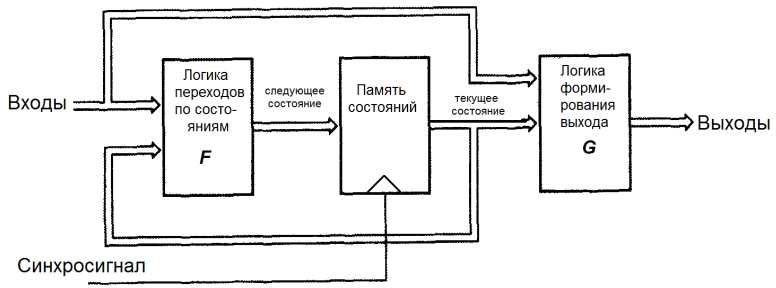


Рис. 1. Структурная схема синхронного конечного автомата Мили

Формальное определение конечного автомата можно дать в следующем виде [1]:

Следующее состояние = *F* (текущее состояние, вход).
Выход = *G*(текущее состояние, вход).

На рис. 2 показан тактируемый синхронный конечный автомат Мили с D-триггерами переключающимися по

положительному фронту. Данная схема выполняет функцию 2-разрядного двоичного синхронного счетчика. Для сравнения посмотрите принципиальную схему синхронного двоичного счетчика на ИС типа SN74163. Логика переходов это есть не что иное, как входные мультиплексоры на входах D-триггеров ИС типа SN74163.

На рис. 3 представлена диаграмма состояний (переходов) конечного автомата на четыре состояния. Схема переходит из состояния в состояние по переднему фронту синхроимпульса при наличие сигнала логической единицы на входе EN. При переходе из состояния 3 в состояние 0 на выходе схемы формируется сигнал MAX (аналог сигнала выхода переноса Cout используемого в счетчиках, позволяет построить счетчики с произвольным модулем счета). Он равен логической единицы, когда равны единицы биты, хранящиеся во всех разрядах счетчика, и подан сигнал разрешения EN.

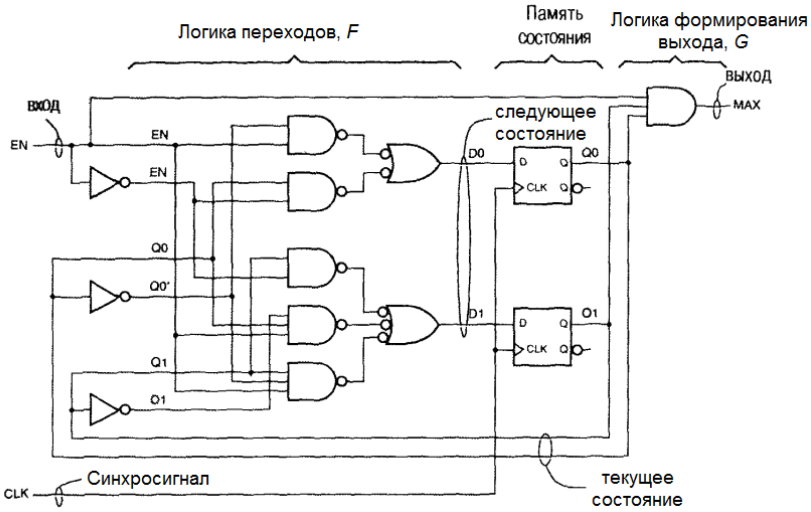


Рис. 2. Тактируемый синхронный конечный автомат Мили с D-триггерами переключающимися по положительному фронту синхросигнала

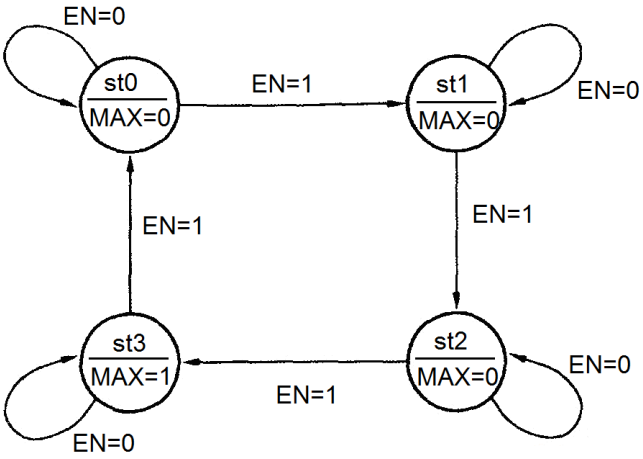


Рис. 3. Диаграмма состояний конечного автомата

Пример проектирования автомата Мили (Mealy) с использованием пакета расширения Stateflow системы Matlab/Simulink и САПР ПЛИС Xilinx ISE Design Suite.

Система Matlab/Simulink содержит встроенный генератор кода языка описания аппаратных средств HDL (Simulink HDL Coder. Simulink HDL Coder – программный продукт для генерации VHDL-кода без привязки к конкретной архитектуре ПЛИС и платформе по Simulink-моделям и граф-автоматам (Stateflow-диаграммы).

Дополнительную информацию по применению системы Matlab/Simulink с пакетом расширения Stateflow и САПР Quartus II для разработки простейшего торгового автомата можно посмотреть в работе [3].

Разработаем имитационную модель 2-разрядного синхронного суммирующего счетчика. В дальнейшем такую схему будем называть конечный автомат. Входной сигнал EN на диаграмме переходов обозначается следующим образом: [EN==1]. Выходной сигнал MAX кодируется следующим образом:

{MAX=0} – счетчик находится в процессе счета;

{MAX=1} – счетчик досчитал до трех, “готов”.

Квадратные скобки [] обозначают условие, фигурные {} – действие по условию. Запись [EN==1]{MAX=0} говорит о том, что выход автомата Мили является функцией, как текущего состояния, так и начального внешнего воздействия, т.е. от сигнала EN.

Автомат может принимать четыре состояния (рис. 4): st0, st1, st2 и st3 в соответствии с рис. 3. На вход автомата EN задается тестовая последовательность из набора единиц. Для того чтобы извлечь VHDL-код из диаграммы переходов необходимо в свойствах диаграммы **Properties** (правый клик мышки) в поле **Execute (enter) Chart At Initialization** поставить галочку (рис. 4г).

После того как, будет создана модель конечного автомата, необходимо выбрать численный метод решения системы дифференциальных уравнений. С помощью проводника модели (Model Explorer) выбираем дискретный метод решения (discrete) дифференциальных уравнений в настройках Solver и настраиваем генератор кода языка VHDL в меню HDL coder. Результат имитационного моделирования показан на рис. 4. На рис. 5 показано имитационное моделирование конечного автомата.

Пример 1 демонстрирует код автомата Мили с регистерным выходом на языке VHDL, полученный с использованием Simulink HDL Coder системы Matlab/Simulink. Анализируя полученный VHDL-код в автоматическом режиме можно сделать вывод, что используется двухпроцессный шаблон описания конечного автомата и перечисляемый тип данных (Enumerated type) T_state_type_is_Chart, который описан в пакете USE work.Avt_pkg (пример 2).

Перечисляемый тип – это такой тип данных, при котором количество всех возможных состояний конечно. Такой тип наиболее часто используется для обозначений состояний конечных автоматов. Любой перечисляемый тип имеет внутреннюю нумерацию: первый элемент всегда имеет номер 0, второй - 1 и т.д. Первый оператор PROCESS со

списком чувствительности (clk, reset) описывает память состояний конечного автомата, а второй оператор PROCESS со списком чувствительности (is_Chart, En, MAX_reg) описывает логику переходов и логику формирования выхода.

Для того чтобы на базе этого кода создать проект в САПР ISE необходимо тип real относящийся к внешним сигналам En, MAX и внутренним MAX_1, MAX_reg, MAX_reg_next заменить на тип std_logic при этом также обеспечить замену 0.0 на '0' и 1.0 на '1' (пример 3). Далее необходимо разработать испытательный стенд (пример 4). На рис.6 показано функциональное моделирование конечного автомата.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.Avt_pkg.ALL;
ENTITY Avt IS
  PORT( clk                 :  IN   std_logic;
        reset               :  IN   std_logic;
        clk_enable          :  IN   std_logic;
        En                  :  IN   real;  -- double
        ce_out              :  OUT  std_logic;
        MAX                 :  OUT  real  -- double
        );
END Avt;
ARCHITECTURE rtl OF Avt IS
  -- Signals
  SIGNAL enb                : std_logic;
  SIGNAL is_Chart           : T_state_type_is_Chart;  -- uint8
  SIGNAL MAX_1              : real := 0.0;  -- double
  SIGNAL MAX_reg            : real := 0.0;  -- double
  SIGNAL is_Chart_next      : T_state_type_is_Chart;  -- enumerated
type (4 enums)
  SIGNAL MAX_reg_next       : real := 0.0;  -- double
BEGIN
  enb <= clk_enable;
  Chart_process : PROCESS (clk, reset)
```

```
BEGIN
  IF reset = '1' THEN
    MAX_reg <= 0.0;
    is_Chart <= IN_st0;
  ELSIF clk'EVENT AND clk = '1' THEN
    IF enb = '1' THEN
      is_Chart <= is_Chart_next;
      MAX_reg <= MAX_reg_next;
    END IF;
  END IF;
END PROCESS Chart_process;
Chart_output : PROCESS (is_Chart, En, MAX_reg)
BEGIN
  is_Chart_next <= is_Chart;
  MAX_reg_next <= MAX_reg;
  CASE is_Chart IS
    WHEN IN_st0 =>
      IF En = 1.0 THEN
        MAX_reg_next <= 0.0;
        is_Chart_next <= IN_st1;
      END IF;
    WHEN IN_st1 =>
      IF En = 1.0 THEN
        MAX_reg_next <= 0.0;
        is_Chart_next <= IN_st2;
      END IF;
    WHEN IN_st2 =>
      IF En = 1.0 THEN
        MAX_reg_next <= 0.0;
        is_Chart_next <= IN_st3;
      END IF;
    WHEN OTHERS =>
      IF En = 1.0 THEN
        MAX_reg_next <= 1.0;
        is_Chart_next <= IN_st0;
      END IF;
  END CASE;
END PROCESS Chart_output;
MAX_1 <= MAX_reg_next;
```

```
  ce_out <= clk_enable;
  MAX <= MAX_1;
END rtl;
```

Пример 1. Код автомата Мили с регистерным выходом на языке VHDL полученный с использованием Simulink HDL Coder системы Matlab/Simulink

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
PACKAGE Avt_pkg IS
  TYPE T_state_type_is_Chart IS (IN_st0, IN_st1, IN_st2, IN_st3);
END Avt_pkg;
```

Пример 2. Пакет, в котором описан перечисляемый тип данных T_state_type_is_Chart

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.Avt_pkg.ALL;
ENTITY Avt IS
  PORT( clk                :  IN   std_logic;
        reset              :  IN   std_logic;
        clk_enable         :  IN   std_logic;
        En                 :  IN   std_logic;
        ce_out             :  OUT  std_logic;
        MAX                :  OUT  std_logic);
END Avt;
ARCHITECTURE rtl OF Avt IS
  -- Signals
  SIGNAL enb                    : std_logic;
  SIGNAL is_Chart               : T_state_type_is_Chart;
  SIGNAL MAX_1                  : std_logic;
  SIGNAL MAX_reg                : std_logic;
  SIGNAL is_Chart_next          : T_state_type_is_Chart;
  SIGNAL MAX_reg_next           : std_logic;
BEGIN
  enb <= clk_enable;
  Chart_process : PROCESS (clk, reset)
```

```
BEGIN
  IF reset = '1' THEN
    MAX_reg <= '0';
    is_Chart <= IN_st0;
  ELSIF clk'EVENT AND clk = '1' THEN
    IF enb = '1' THEN
      is_Chart <= is_Chart_next;
      MAX_reg <= MAX_reg_next;
    END IF;
  END IF;
END PROCESS Chart_process;
Chart_output : PROCESS (is_Chart, En, MAX_reg)
BEGIN
  is_Chart_next <= is_Chart;
  MAX_reg_next <= MAX_reg;
  CASE is_Chart IS
    WHEN IN_st0 =>
      IF En = '1' THEN
        MAX_reg_next <= '0';
        is_Chart_next <= IN_st1;
      END IF;
    WHEN IN_st1 =>
      IF En = '1' THEN
        MAX_reg_next <= '0';
        is_Chart_next <= IN_st2;
      END IF;
    WHEN IN_st2 =>
      IF En = '1' THEN
        MAX_reg_next <= '0';
        is_Chart_next <= IN_st3;
      END IF;
    WHEN OTHERS =>
      IF En = '1' THEN
        MAX_reg_next <= '1';
        is_Chart_next <= IN_st0;
      END IF;
  END CASE;
END PROCESS Chart_output;
MAX_1 <= MAX_reg_next;
```

```
  ce_out <= clk_enable;
  MAX <= MAX_1;
END rtl;
```

Пример 3. Отредактированный код автомата Мили на языке VHDL полученный с использованием Simulink HDL Coder системы Matlab/Simulink

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY avt_sch_avt_sch_sch_tb IS
END avt_sch_avt_sch_sch_tb;
ARCHITECTURE behavioral OF avt_sch_avt_sch_sch_tb IS
  COMPONENT avt_sch
  PORT( clk     :       IN      STD_LOGIC;
        ares    :       IN      STD_LOGIC;
        clk_ena :       IN      STD_LOGIC;
        en      :       IN      STD_LOGIC;
        ce_out  :       OUT     STD_LOGIC;
        MAX     :       OUT     STD_LOGIC);
  END COMPONENT;
  SIGNAL clk  :       STD_LOGIC;
  SIGNAL ares :       STD_LOGIC:= '0';
  SIGNAL clk_ena      :       STD_LOGIC:= '1';
  SIGNAL en   :       STD_LOGIC:= '1';
  SIGNAL ce_out       :       STD_LOGIC;
  SIGNAL MAX          :       STD_LOGIC;
          -- Clock period definitions
  constant clk_period : time := 100 ns;
BEGIN
  UUT: avt_sch PORT MAP(
                clk => clk,
                ares => ares,
                clk_ena => clk_ena,
                en => en,
                ce_out => ce_out,
```

```
                MAX => MAX
  );
  -- Clock process definitions
  clk_process :process
  begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
  end process;
   tb : process
                begin
                        wait for 100 ns;
                        en <= '1';
                        wait for 100 ns;
                        en <= '1';
                        wait for 100 ns;
                        en <= '1';
                        wait for 100 ns;
                        en <= '1';
          wait;
  end process;

END;
```

Пример 4. Испытательный стенд автомата Мили

Вопросы для отчета

1. Дайте определение автомата Мили, Мура?
2. Приведите структурная схема синхронного конечного автомата Мили?
3. Какие функции выполняет цифровой автомат?
4. Методы кодирования автоматов?
5. Нарисуйте схему счетчика-делителя на три с двоичным кодированием

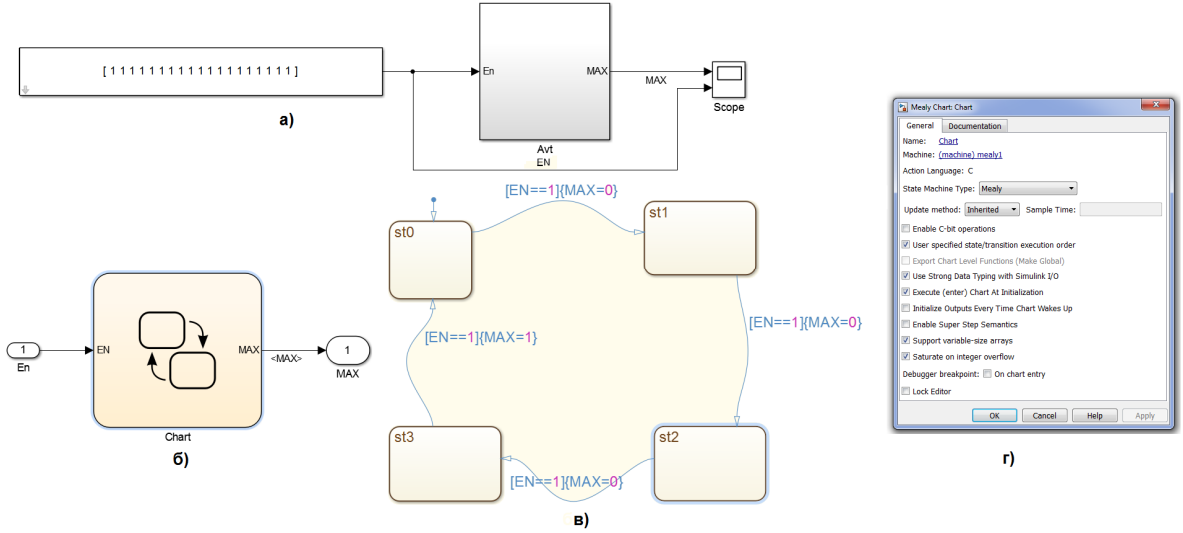


Рис. 4. Модель конечного автомата Мили в системе Matlab/Simulink с использованием Stateflow-диаграммы: а) верхний уровень; б) нижний уровень иерархии; в) диаграмма переходов, разработанная с помощью пакета расширения Stateflow; г) свойства диаграммы

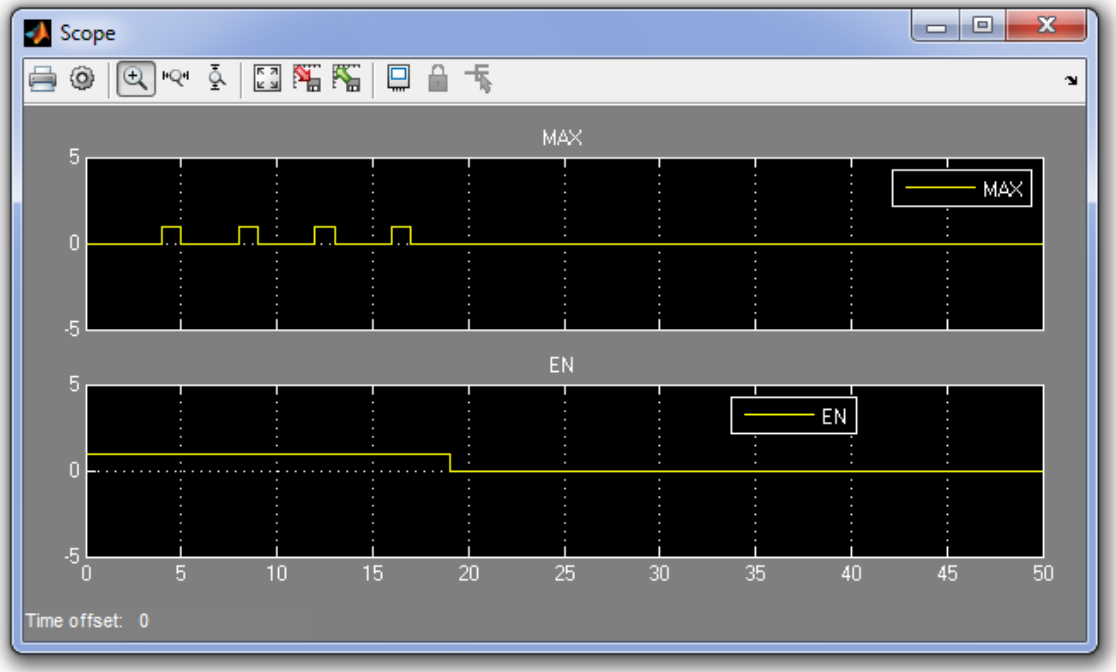


Рис. 5. Имитационное моделирование конечного автомата Мили в системе Matlab/Simulink

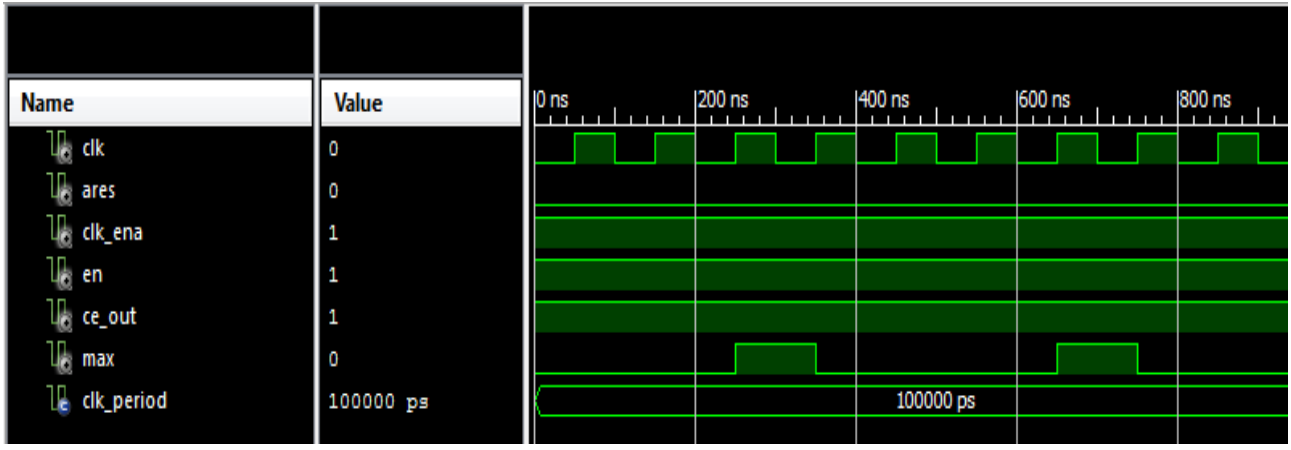


Рис. 6. Функциональное моделирование конечного автомата Мили в САПР Xilinx ISE

ЛАБОРАТОРНАЯ РАБОТА № 2

Проектирования автомата Мура с использованием методологии Black Boxes Xilinx System Generator

Цель работы: освоение методологии Black Boxes Xilinx System Generator используемой для разработки цифровых автоматов

Задание:

1) используя одно, двух – и трехпроцессные шаблоны разработайте VHDL-код конечного автомата согласно выбранному заданию с применением атрибута syn_encoding "onehot";

2) реализуйте функциональную модель конечного автомата в любом из доступных вам САПР (САПР Altera Quartus II или Xilinx ISE) и убедитесь в правильности ее функционирования;

3) разработайте имитационную модель конечного автомата в Matlab/Simulink с использованием единственного VHDL-файла и методологии Black Boxes Xilinx System Generator;

4) осуществите имитационное моделирование и убедитесь в правильности функционирования конечного автомата;

5) извлеките в автоматическом режиме VHDL-код и осуществите функциональное моделирование в Xilinx ISE

Теоретические сведения

Рассмотрим трехпроцессный шаблон описания работы конечного автомата Мура (рис. 7, пример 5). В данном случае используются два сигнала state и next_state перечисляемого типа и три оператора Process. Проект разработаем в САПР ПЛИС Altera Quartus II v13.1 Build 162 с использованием ModelSim Altera Started Edition 10.1 d. Для кодирования состояний цифрового автомата будем использовать метод one

hot encoding (OHE - кодирование с одним активным, или горячим состоянием или унитарное кодирование). Кодирование по методу OHE будет определено явно в коде языка VHDL с использованием атрибута syn_encoding "onehot" [2,5].

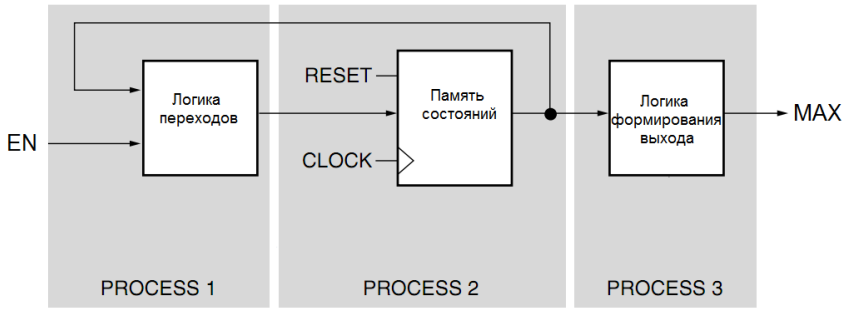


Рис. 7. Трехпроцессный шаблон описания работы конечного автомата Мура на языке VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY counter2 IS
        PORT(
        en,res,clk,ce   : IN    STD_LOGIC;
        Max     : OUT STD_LOGIC
        );
END counter2;
ARCHITECTURE a OF counter2 IS
TYPE state_values IS (Stage0, Stage1, Stage2, Stage3);
signal state, next_state: state_values;
attribute syn_encoding: string;
attribute syn_encoding of state, next_state: signal is "onehot";
BEGIN
statereg: process(clk,ce,res)
begin
        if res = '1' then state<=Stage0;
```

```
                elsif (clk'event and clk='1') then
                        if ce = '1' then state<=next_state;
                        end if;
        end if;
end process statereg;
process(state, en)
begin
case state is
when Stage0=>
IF en='1' THEN next_state<=Stage1;
else next_state<=Stage0;
END IF;
when Stage1=>
IF en='1' THEN next_state<=Stage2;
else next_state<=Stage1;
END IF;
when Stage2=>
IF  en='1' THEN next_state<=Stage3;
else next_state<=Stage2;
END IF;
when Stage3=>
IF en='1' THEN next_state<=Stage0;
else next_state<=Stage3;
END IF;
when others=>
next_state<=Stage0;
end case;
end process;
process (state)
begin
case state is
when Stage0=>
MAX  <= '0';
when Stage1=>
MAX  <= '0';
when Stage2=>
```

```
MAX  <= '0';
when Stage3=>
MAX  <= '1';
end case;
end process;
END a;
```

Пример 5. Трехпроцессный шаблон описания конечного автомата на языке VHDL использованием атрибута syn_encoding "onehot"

Метод получил такое название потому, что в каждый конкретный момент времени активным (hot) может быть только один триггер состояния. Применение метода ОНЕ для ПЛИС по архитектуре ППВМ наиболее перспективно [4,5]. Метод ОНЕ применительно к ПЛИС по архитектуре ППВМ, дает возможность строить конечные автоматы, которые в общем случае требуют меньших ресурсов и отличаются более высокими скоростными показателями, чем аналогичные конечные автоматы с двоичным кодированием состояний. Повышенное быстродействие по методу ОНЕ обеспечивается меньшим числом уровней логики между рабочими фронтами синхросигналов, чем в случае двоичного кодирования. Логические схемы при этом упрощаются, поскольку метод ОНЕ практически не требует логики декодирования состояний. Получающийся в результате построения конечного автомата набор триггеров похож на структуру типа сдвигового регистра.

Быстродействие конечного автомата типа ОНЕ остается постоянным с увеличением числа состояний. И напротив, быстродействие конечного автомата с высокой степенью кодирования состояний снижается с увеличением количества состояний, поскольку в этом случае для декодирования требуется большее число уровней логики с большим числом линий.

По умолчанию в настройках САПР Quartus II категория Analysis & Synthesis Settings /More Settings в разделе State

Machine Processing задан режим Auto (рис. 8а). Отменим автоматическое кодирование перечисления и определим свои собственные коды с помощью атрибута enum_encoding (пример 5). Атрибут enum_encoding является строкой (string), содержащей набор векторов, по одному для каждого перечисляемого литерала в соответствующем типе. Первый вектор в строке атрибута определяет код для первого перечисляемого литерала, второй вектор - для второго литерала и т.д. Атрибут enum_encoding должен следовать сразу же за объявлением типа. Рис. 8б показывает восстановленную из кода языка VHDL диаграмму переходов, а рис.8в демонстрирует таблицу переходов соответствующую методу ОНЕ. Полученный в результате синтеза конечный автомат с кодированием по методу ОНЕ имеет структуру похожую на сдвиговый регистр (рис. 8г). На рис. 9 показано функциональное моделирование конечного автомата в симуляторе ModelSim Altera Started Edition и в векторном редакторе САПР Quartus II. Демонстрируются изменения на сигналах state и next_state в процессе перехода по состояниям.

Убедившись с помощью временных диаграмм (рис. 9) в правильности функционирования конечного автомата разработаем имитационную модель (рис. 10) в Matlab/Simulink с использованием единственного VHDL-файла и методологии Black Boxes Xilinx System Generator. Имитационное моделирование конечного автомата в системе Matlab/Simulink с применением VHDL-кода (функциональный блок Black Box) показано на рис. 11.

С помощью маркера Xilinx System Generator извлечем в автоматическом режиме оптимизированный VHDL-код и осуществим функциональное моделирование (рис. 12). С помощью рис. 11 и 12 убеждаемся в правильности работы конечного автомата.

Предварительно удостоверимся, что в настройках компилятора ПЛИС Xilinx ISE Design Suite для синтеза конечного автомата по умолчанию выбран стиль кодирования avto (рис. 13). Рис. 14 показывает, что компилятор ПЛИС

Xilinx ISE распознает атрибут syn_encoding "onehot" для сигналов state и next_state перечисляемого типа и применяет метод кодирования onehot. Однако таблица переходов отличается от той, что используется в САПР Altera Quartus II (рис. 8 б). В САПР Quartus II используется инверсия младшего разряда таблицы переходов, т.е. в состоянии отличного от нулевого (начальное, st0) первый триггер в цепочке сдвигового регистра всегда активен.

Получаемая в результате синтеза структура сдвигового регистра в основе автомата с кодированием по методу ОНЕ на этапе размещения в технологический базис ПЛИС Spartan-6 xc6slx4-3tqg144 показана на рис. 15.

Вопросы для отчета

1. Из каких составных частей состоит конечный автомат?
2. Дайте определение конечного автомата Мили, Мура ?
3. Приведите структурную схема синхронного конечного автомата Мили ?
4. Какие функции выполняет цифровой автомат ?
5. Перечислите методы кодирования автоматов ?
6. Нарисуйте схему проектируемого автомата с кодированием по методу ОНЕ.

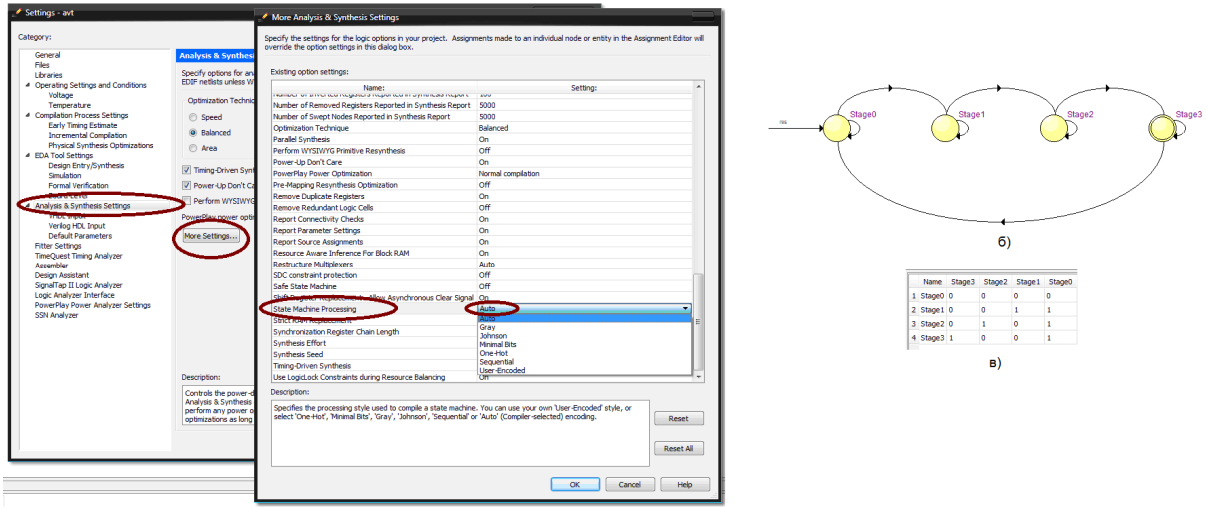
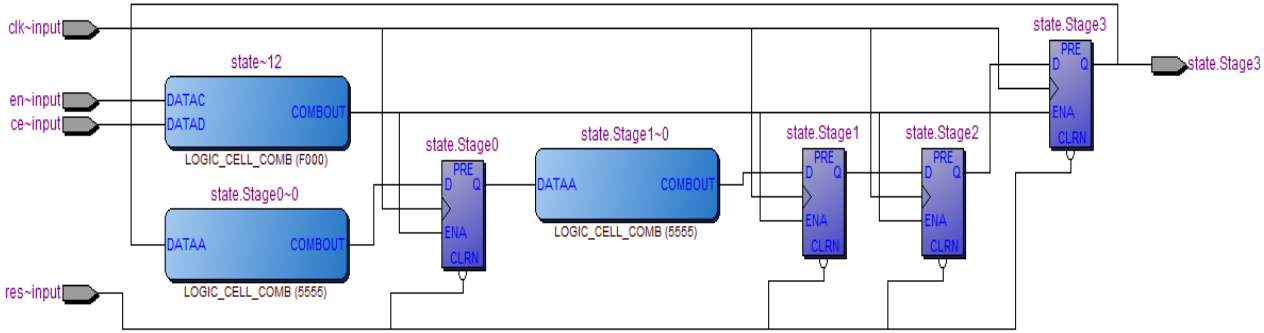


Рис. 8. Синтез конечного автомата по VHDL-коду в САПР Altera Quartus II (трехпроцессный шаблон): а) настройки компилятора САПР Quartus II для синтеза конечного автомата (по умолчанию выбран стиль кодирования avto); б) граф-автомат восстановленный из VHDL-кода; в) таблица переходов, демонстрирующая использование метода ОНЕ; г) структура сдвигового регистра в основе автомата с кодированием по методу ОНЕ на этапе размещения в технологический базис ПЛИС



г)

Рис. 8. Синтез конечного автомата по VHDL-коду в САПР Altera Quartus II (трехпроцессный шаблон): а) настройки компилятора САПР Quartus II для синтеза конечного автомата (по умолчанию выбран стиль кодирования avto); б) граф-автомат восстановленный из VHDL-кода; в) таблица переходов, демонстрирующая использование метода ОНЕ; г) структура сдвигового регистра в основе автомата с кодированием по методу ОНЕ на этапе размещения в технологический базис ПЛИС (продолжение)

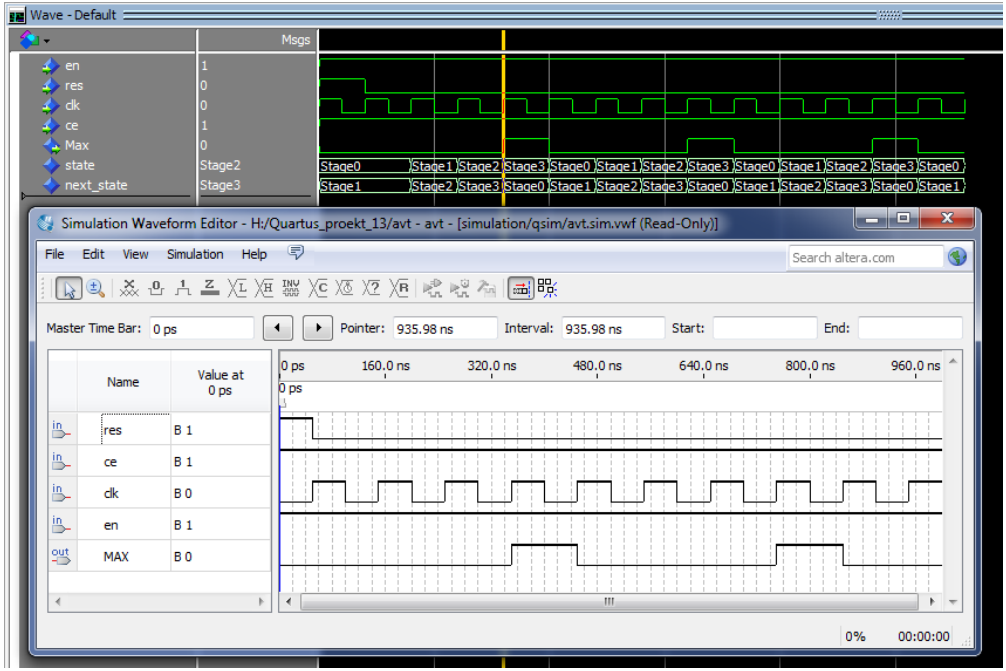


Рис. 9. Функциональное моделирование конечного автомата в ModelSim Altera Started Edition и в векторном редакторе САПР Altera Quartus II

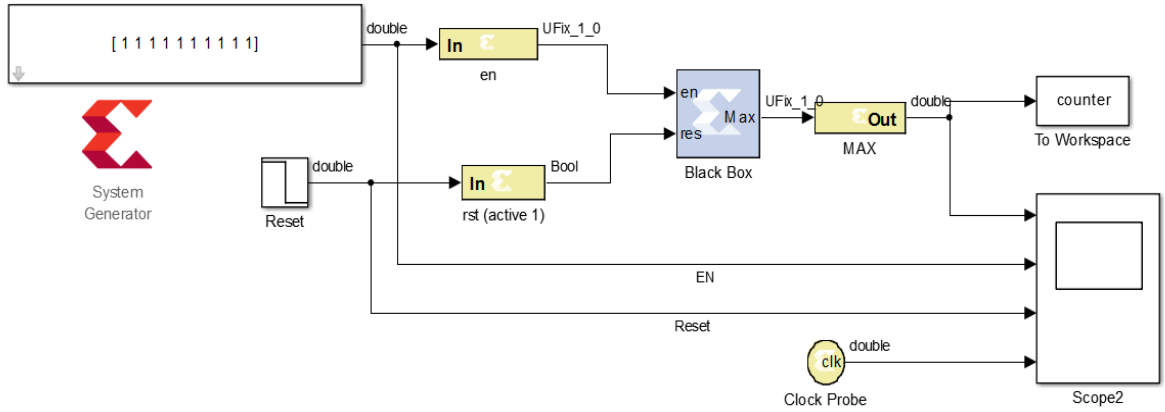


Рис. 10. Имитационная модель конечного автомата на основе функционального блока Black Box с использованием единственного VHDL-файла в системе Matlab/Simulink

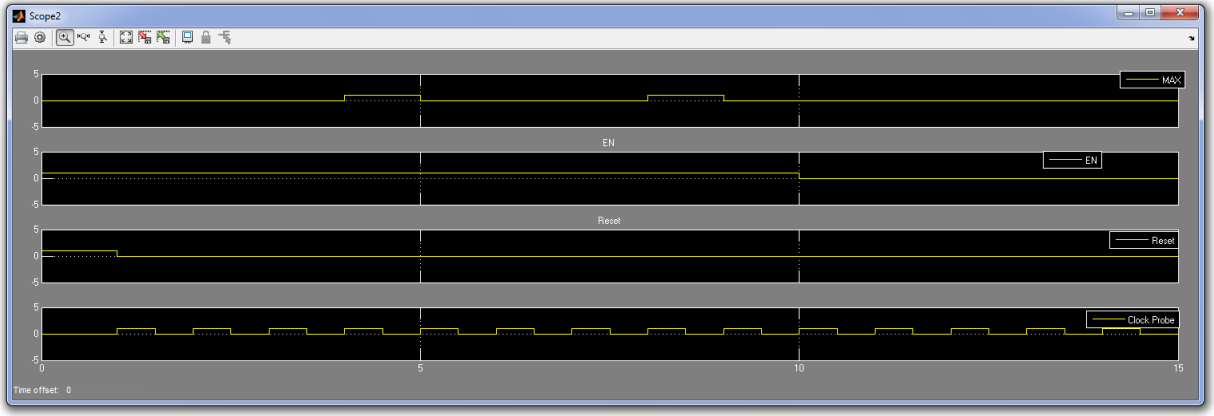


Рис. 11. Имитационное моделирование конечного автомата в системе Matlab/Simulink с использованием единственного VHDL-файла (функциональный блок Black Box)

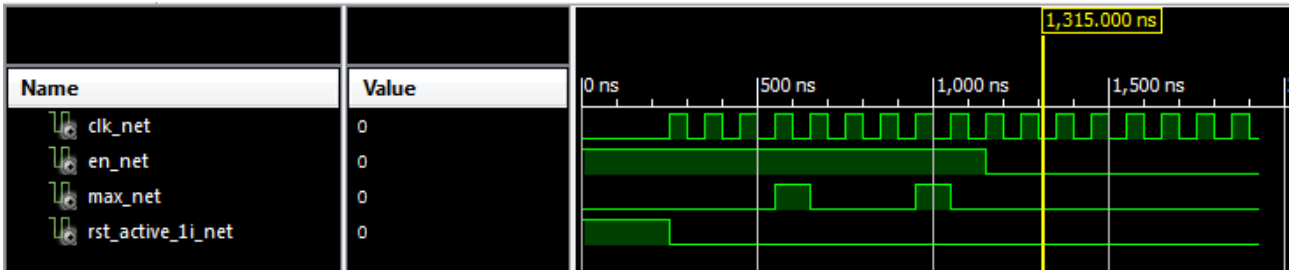


Рис. 12. Функциональное моделирование конечного автомата в САПР Xilinx ISE

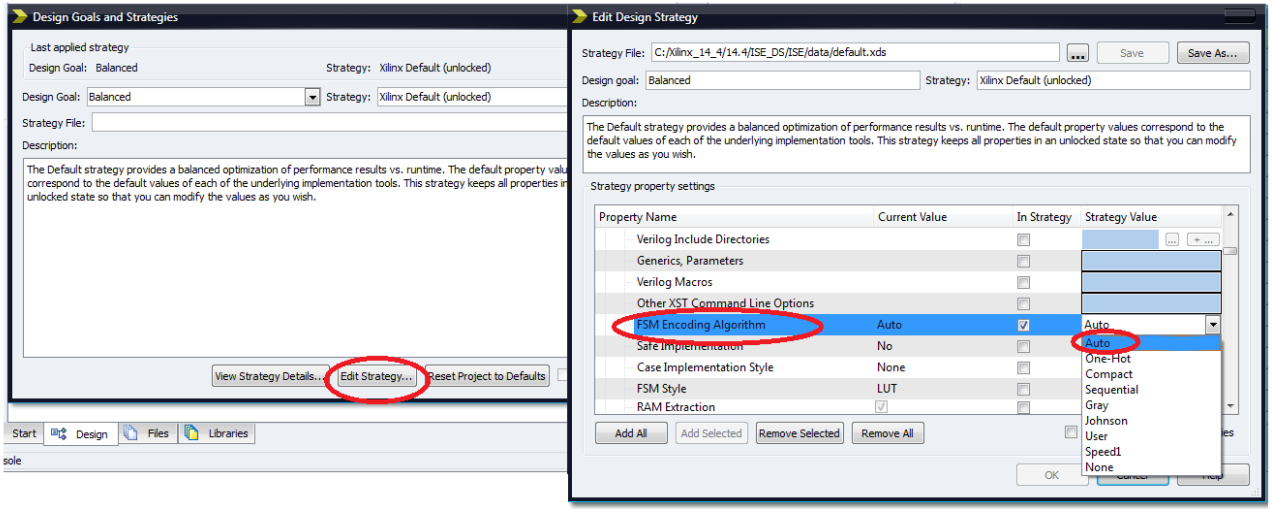


Рис. 13. Настройки компилятора ПЛИС Xilinx ISE Design Suite для синтеза конечного автомата (по умолчанию выбран стиль кодирования avto)

```
Set property "syn_encoding = onehot" for signal <state>.
Set property "syn_encoding = onehot" for signal <next_state>.
Found 2-bit register for signal <state>.
Found finite state machine <FSM_0> for signal <state>.
-----------------------------------------------------------------------
| States             | 4                                              |
| Transitions        | 12                                             |
| Inputs             | 2                                              |
| Outputs            | 1                                              |
| Clock              | clk (rising_edge)                              |
| Reset              | res (positive)                                 |
| Reset type         | asynchronous                                   |
| Reset State        | stage0                                         |
| Power Up State     | stage0                                         |
| Encoding           | one-hot                                        |
| Implementation     | LUT                                            |
-----------------------------------------------------------------------

=======================================================================
*                          Low Level Synthesis                          *
=======================================================================
Optimizing FSM <FSM_0> on signal <state[1:4]> with one-hot encoding.
--------------------
 State  | Encoding
--------------------
 stage0 | 0001
 stage1 | 0010
 stage2 | 0100
 stage3 | 1000
--------------------
```

Рис. 14. Таблица переходов, демонстрирующая использование метода ONE

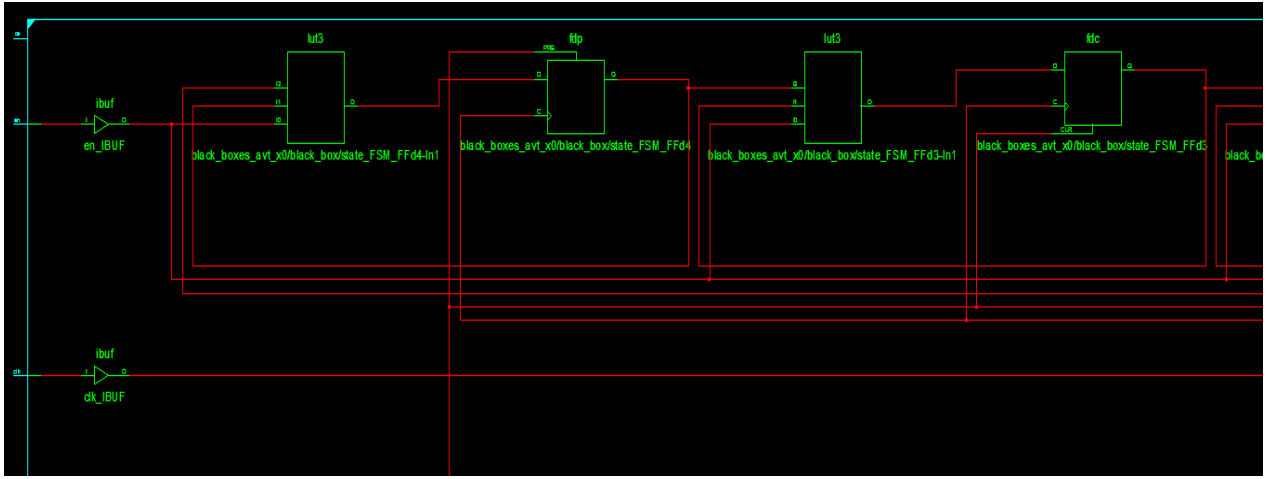


Рис. 15. Структура сдвигового регистра в основе автомата с кодированием по методу ОНЕ на этапе размещения в технологический базис ПЛИС Spartan6 xc6slx4-3tqg144

ЛАБОРАТОРНАЯ РАБОТА № 3
Проектирование конечных автоматов с использованием функциональных блоков из библиотеки Reference Blockset/Control Logic Xilinx System Generator

Цель работы: освоение методологии Black Boxes Xilinx System Generator для разработки цифровых автоматов

Задание:

1) разработайте согласно выбранному варианту задания имитационную модель конечного автомата на основе функционального блока Mealy State Machine из библиотеки Reference Blockset/Control Logic;

3) осуществите имитационное моделирование и убедитесь в правильности функционирования конечного автомата;

4) извлеките в автоматическом режиме VHDL-код и осуществите функциональное моделирование в Xilinx ISE;

5) дайте оценку быстродействия проекта и задействованных логических ресурсов ПЛИС.

Теоретические сведения

На рис. 16 показано расположение функционального блока Mealy State Machine в библиотеке Xilinx Reference Blockset/Control Logic. На основе этого блока разработаем имитационную модель (рис. 17а) и проведем моделирование (рис. 17б). Предварительно необходимо отредактировать матрицу переходов и матрицу формирования выходного сигнала (рис. 18а) согласно рис. 3. Задать разрядность выходного сигнала (1-разрядный выходной сигнал MAX) и время дискретизации (1 с). На рис. 18б показана структурная схема автомата Мили, а на рис. 19 представлено функциональное моделирование в САПР Xilinx ISE.

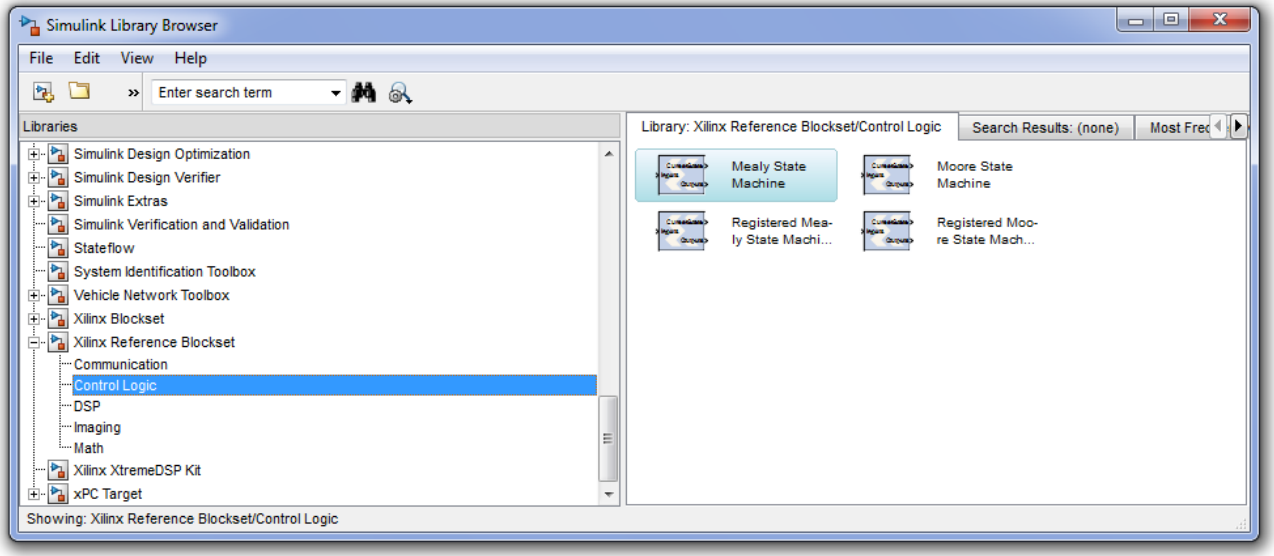
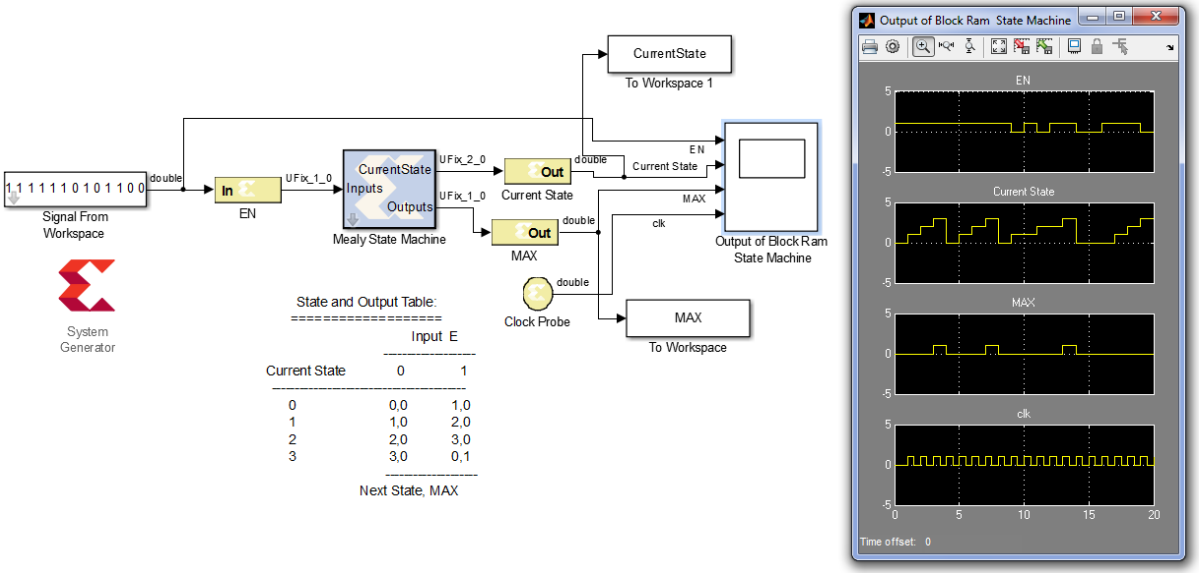
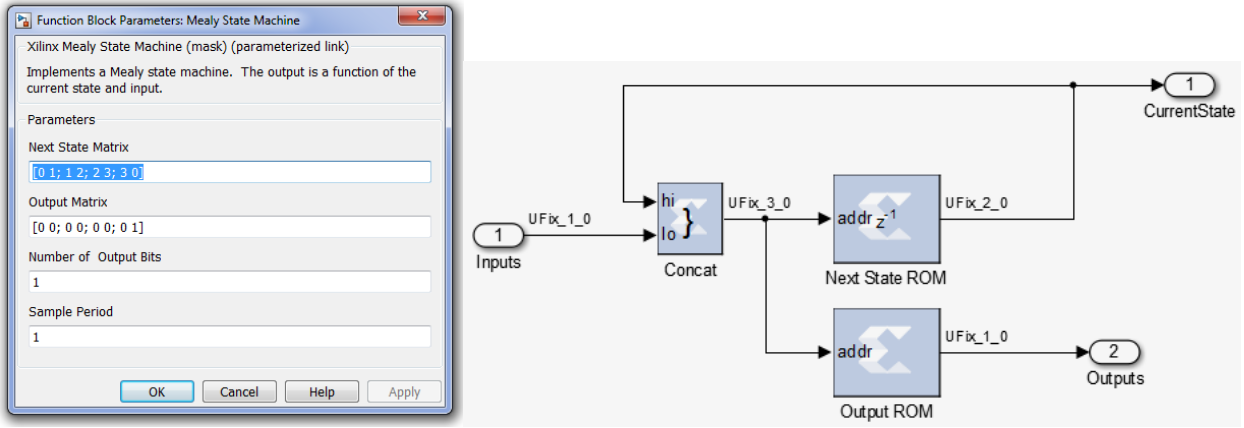


Рис. 16. Расположение функционального блока Mealy State Machine в библиотеке Xilinx Reference Blockset/Control Logic



а) б)

Рис. 17. а) Имитационная модель конечного автомата на основе функционального блока Mealy State Machine представленного на рис.3; б) результаты моделирования



а) б)

Рис. 18. а) Настройки функционального блок Mealy State Machine; б) структурная схема автомата Мили

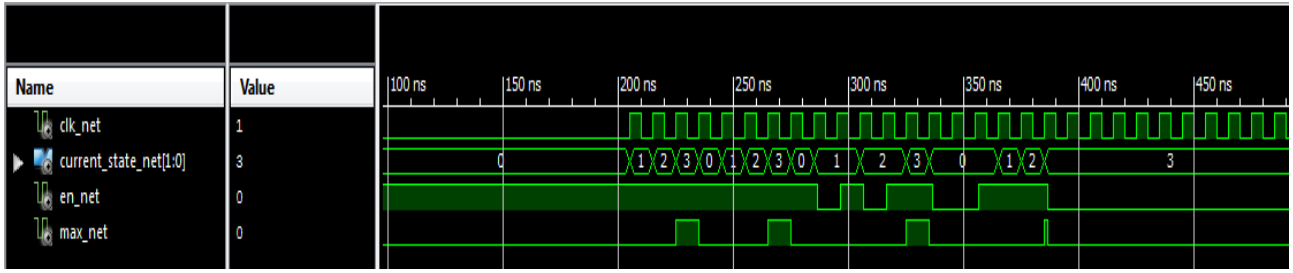


Рис. 19. Функциональное моделирование конечного автомата на основе функционального блока Mealy State Machine в САПР Xilinx ISE

System Generator в автоматическом режиме создает файл проектных ограничений по распространению сигналов, например, исходя из требования к периоду синхросигнала (10 нс) указанного в настройках маркера System Generator закладка Clocking поле FPGA clock period. Параметр PERIOD задает ограничение на максимальное время распространения сигнала от выхода одного синхронного элемента до входа другого. Максимальная рабочая частота проекта при заданных временных ограничениях для ПЛИС Spartan-6 xc6slx4-3tqg144 на этапе post-map составляет 320 МГц. Для сравнения, кодирование автомата по методу ОНЕ при тех же проектных ограничениях (NET "clk" TNM_NET = "clk_7796afde"; TIMESPEC "TS_clk_7796afde" = PERIOD "clk_7796afde" 10.0 ns HIGH 50 %;) позволяет получить рабочую частоту 578 МГц.

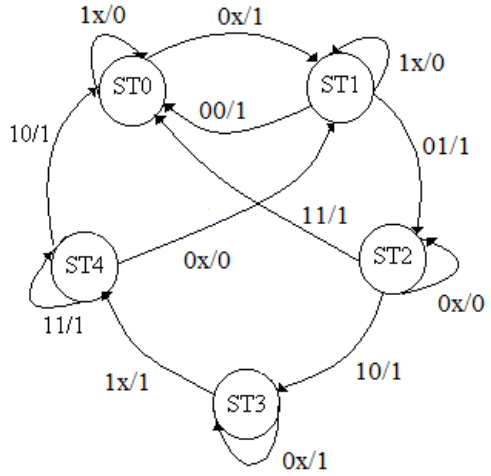
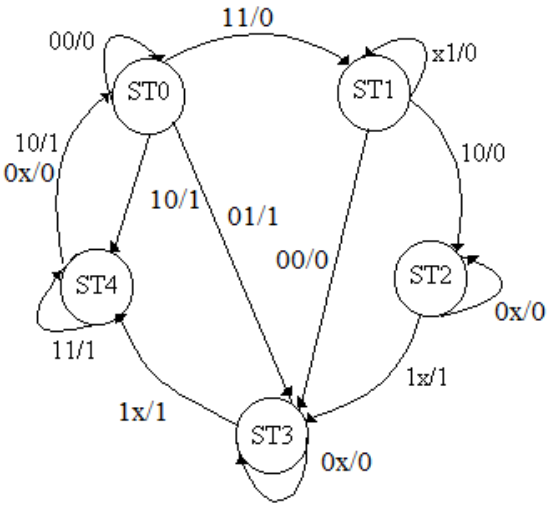
Использование объектно-ориентированного проектирования предоставляемого Xilinx System Generator в совокупностью с методологией Black Boxes и с функциональными блоками из библиотеки Reference Blockset/Control Logic позволяет обеспечить не только имитационное моделирование в системе Matlab/Simulink, но и осуществлять ускоренное проектирование конечных автоматов в САПР Xilinx ISE с гарантируемым быстродействием.

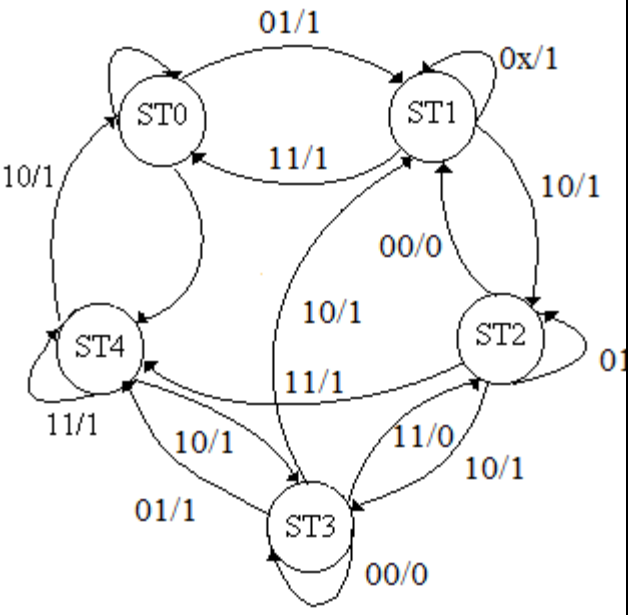
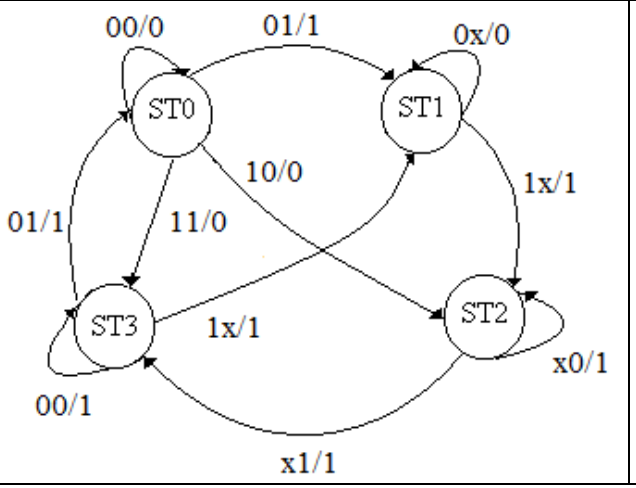
Вопросы для отчета

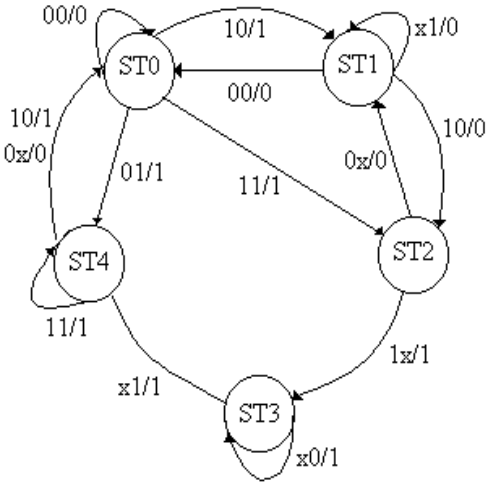
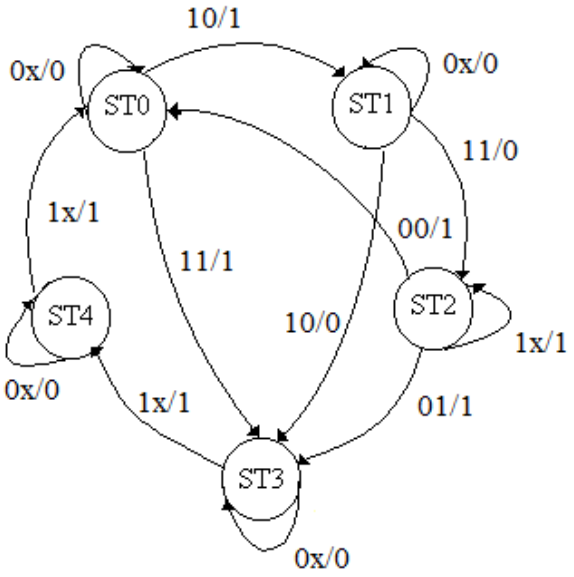
1. Приведите пример конечного автомата с двумя входами и выходами. Постройте для него таблицу переходов, осуществите кодирование состояний и выходов?
2. Приведите структурную схему синхронного конечного автомата Мили с двумя входами и выходами?
3. Приведите пример таблицы переходов и выходов автомата Мили с кодированием состояний?
4. Восстановите автомат по электрической схеме?
5. Нарисуйте схему проектируемого автомата с кодированием по методу ОНЕ.

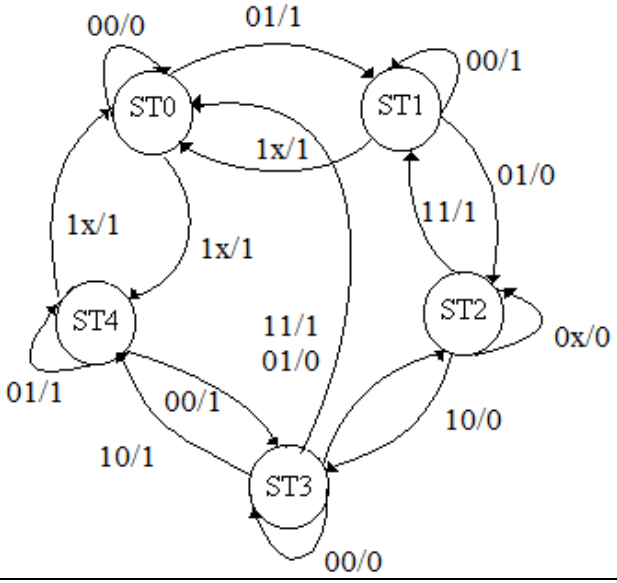
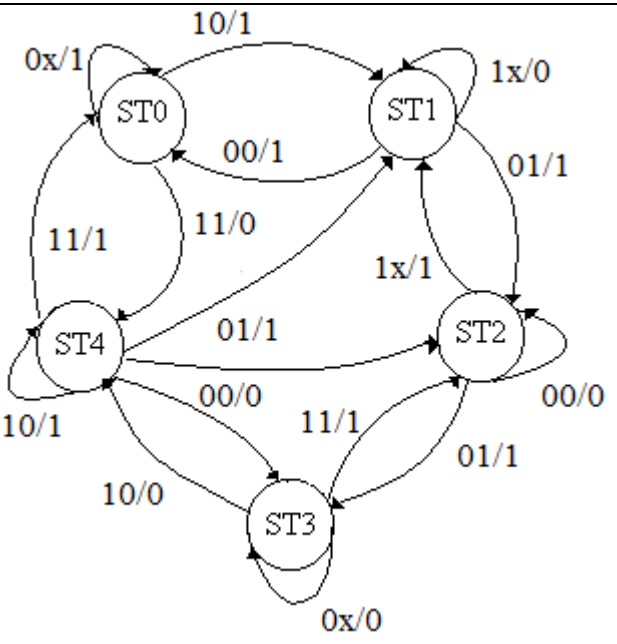
ПРИЛОЖЕНИЕ

Варианты заданий для выполнения лабораторных работ

	Вариант задания	ФИО студента
1.		
2.		

3.	 01/1 0x/1 ST0 ST1 11/1 10/1 10/1 00/0 10/1 ST4 ST2 11/1 11/1 10/1 11/0 10/1 01/1 ST3 00/0	
4.	 00/0 01/1 0x/0 ST0 ST1 10/0 1x/1 01/1 11/0 ST3 1x/1 ST2 x0/1 00/1 x1/1	

5.	 00/0 10/1 x1/0 ST0 ST1 00/0 10/1 0x/0 10/0 01/1 11/1 0x/0 ST4 ST2 11/1 1x/1 x1/1 ST3 x0/1	
6.	 10/1 0x/0 0x/0 ST0 ST1 11/0 1x/1 00/1 11/1 ST4 ST2 10/0 1x/1 0x/0 1x/1 01/1 ST3 0x/0	

7.	 00/0 01/1 00/1 ST0 ST1 1x/1 01/0 11/1 1x/1 1x/1 ST4 ST2 0x/0 11/1 01/0 01/1 00/1 10/0 10/1 ST3 00/0	
8.	 10/1 0x/1 1x/0 ST0 ST1 00/1 01/1 11/1 11/0 1x/1 ST4 01/1 ST2 00/0 00/0 10/1 11/1 01/1 10/0 ST3 0x/0	

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Уэйкерли Джон Ф. Проектирование цифровых устройств: пер. с англ. / Ф. Джон Уэйкерли. М.: Постмаркет, 2002. 533 с.

2. Тарасов И. Методы и программные продукты для повышения производительности проектов на базе ПЛИС Xilinx. [Текст] / И.В. Тарасов // Компоненты и технологии. 2008. N2. С.88-93.

3. Строгонов А.В. Проектирование цифровых автоматов с использованием системы MATLAB/Simulink II. [Текст] / А.В. Строгонов // Компоненты и технологии. 2008. N4. С.32-36.

4. Строгонов А.В. Проектирование конечных автоматов по методу ОНЕ. [Текст] / А.В. Строгонов // Компоненты и технологии. 2007. N10. С.82-87.

5. Строгонов А.В., Быстрицкий А.В. Эффективность разработки конечных автоматов в базисе ПЛИС FPGA. II [Текст] / А.В. Строгонов, А.В. Быстрицкий // Компоненты и технологии. 2013. N1. С.66-72.

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ
№ 1-3 по дисциплине «Проектирование цифровых устройств в базисе ПЛИС» для студентов направления 11.03.04 «Электроника и наноэлектроника» (направленность «Микроэлектроника и твердотельная электроника») очной формы обучения

Составители:
Строгонов Андрей Владимирович
Кошелева Наталья Николаевна
Арсентьев Алексей Владимирович
Винокуров Александр Александрович

В авторской редакции

Компьютерный набор А.В. Строгонова

Подписано к изданию 26.10.2016.
Уч.-изд. л. 2,4.

ФГБОУ ВО “Воронежский государственный технический университет”
394026 Воронеж, Московский просп., 14