Проектирование цифровых **автоматов** с использованием системы MATLAB/Simulink

Андрей СТРОГОНОВ,
д. т. н.
andreis@hotmail.ru

Цель работы — демонстрация возможностей системы MATLAB/Simulink (пакет расширения Stateflow) по проектированию цифровых автоматов [1], представленных графом переходов, с последующей их реализацией в базисе ПЛИС с использованием САПР Quartus и симулятора ModelSim (Mentor Graphics HDL simulator).

Simulink — графическая среда имитационного моделирования аналоговых и дискретных систем. Она предоставляет пользователю графический интерфейс для конструирования моделей из стандартных блоков, без единой строчки кода. Simulink работает с линейными, нелинейными, непрерывными, дискретными и многомерными системами. Система MATLAB/Simulink содержит встроенный генератор кода языка описания аппаратных средств HDL (Simulink HDL Coder) и ориентирована на поддержку симулятора VHDL ModelSim. 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). Система MATLAB/Simulink эффективна также при проектировании цифровых фильтров для реализации в базисе ПЛИС и ЦОС процессоров, так как содержит Filter Design HDL Coder.

ModelSim — наиболее распространенный в мире VHDL и VHDL/Verilog-симулятор. Популярность ModelSim отражает стремление фирмы Mentor Graphics предоставить пользователям самую передовую технологию моделирования, высокую производительность и полную техническую поддержку. Семейство ModelSim имеет уникальную архитектуру, основанную на принципе «оптимизированной прямой компиляции» и «едином ядре моделирования».

Архитектура, базирующаяся на принципе оптимизированной прямой компиляции, является технологией нового поколения в области HDL-моделирования. Она составляет основу всех продуктов семейства ModelSim. В соответствии с этим принципом исходный VHDL- или Verilog-код компилируется в машинно-независимый объектный код, исполняемый на любой поддерживаемой платформе (САПР БИС или ПЛИС). Непосредственно скомпилированные, HDL-объекты автоматически оптимизируются для любой поддерживаемой платформы в момент запуска программы ModelSim.

Многие САПР БИС, например, Mentor Graphics (HDL Designer) [2, 3] и САПР ПЛИС, такие как Foundation фирмы Xilinx (система синтеза FPGA Express Synthesis, разработанная компанией Synopsys) [4], StateCAD фирмы Visual Software Solutions [5], Quartus II (версия 7.2) фирмы Altera, также содержат встроенные средства проектирования цифровых автоматов, позволяют задавать цифровой автомат графом переходов [6, 7] и получать автоматически код языка VHDL или Verilog. Общий недостаток — их узкая направленность (проектирование заказных БИС или разработка цифровых устройств в базисе ПЛИС на функциональном и логическом уровнях проектирования).

В настоящее время систему MATLAB/Simulink активно используют на системном уровне проектирования (Electronic System Level, ESL). ESL-проектирование, или проектирование «сверху вниз», основано на имитационном моделировании. Ключ к реализации ESL-подхода — моделирование на более высоких уровнях абстракции. Проще и быстрее разрабатывать модели на более высоких уровнях абстракции, нежели на уровне регистровых передач (RTL-уровень, или разновидность стиля языка HDL). В целом ESL-проектирование направлено на сокращение сроков проектирования. Идеальной выглядит методология последовательного перехода на более низкие уровни абстракции: от функциональной к RTL-модели [8].

На системном уровне проектирования проводится верификация алгоритмов, как правило, с помощью языка программирования C/C++ или специальных средств типа MATHLAB/Simulink или SPW (Signal Processing WorkSystems фирмы Cadence Design Systems), с привлечением радиочастотных, коммуникационных, мультимедийных и других библиотек. В процессе выполнения работ на системном уровне формируется описание аппаратной части проекта в RTL-кодах, что существенно упрощает процесс подготовки технического задания для этапов функционального и логического проектирования. Summit Design и CoWare (Cadence Design Systems) — одни из самых сильных компаний, производителей средств ESL-проектирования.

Рассмотрим проектирование автомата Мили (Mealy) с использованием системы MATLAB/Simulink и САПР ПЛИС Quartus. На рис. 1а показан испытательный стенд (модель) автомата Мили в системе MATLAB/Simulink. Пример автомата Мили позаимствуем из справочной системы Simulink. Торговый автомат (рис. 1б) предназначен для выдачи бутылки сладкой шипучей жидкости (сигнал soda), когда опущено 15 центов или более. Торговый автомат не совершенен и сдачи не дает, то есть оставляет «себе» монету в 5 центов, которая будет добавлена к общему вкладу. Пример более совершенного торгового автомата можно найти в книге известных американских специалистов Хоровица и Хилла «Искусство схемотехники».

Существует некоторый вид монетного интерфейса, который «заглатывает», распознает монету и посылает на входы автомата сигнал Coin (монета). Монетный интерфейс реализуется с использованием сигнала Coin (рис. 1в). Аналоговый входной сигнал Coin на диаграмме переходов кодируется следующим образом: [Coin= =1] — брошена монета в 5 центов (nickel); [Coin= =2] — брошена монета в 10 центов (dime)), где 1, 2 — переменные вещественного типа. Поэтому сигнал Coin должен принимать значения 1 или 2 (рис. 1в). Выходной сигнал Soda кодируется следующим образом:

- {Soda=0} — нет бутылки;
- {Soda=1} — бутылка.

Квадратные скобки [] обозначают условие, фигурные {} — действие по условию. Запись [Coin= =1]{Soda=0} говорит о том, что выход автомата Мили является функцией как текущего состояния, так и начального внешнего воздействия, то есть сигнала Coin.

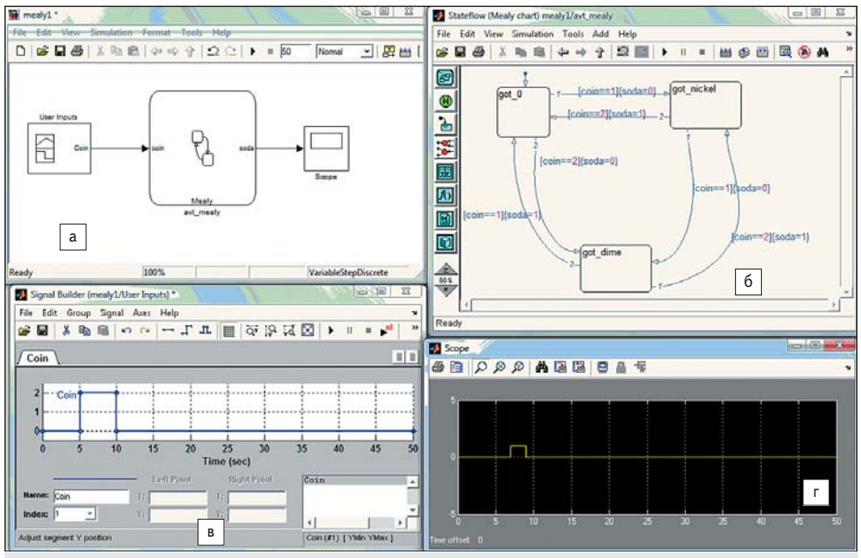

Рис. 1. Автомат Мили, построенный с использованием системы MATLAB/Simulink:
а) испытательный стенд; б) граф-автомат; в) входной сигнал Coin; г) выходной сигнал Soda

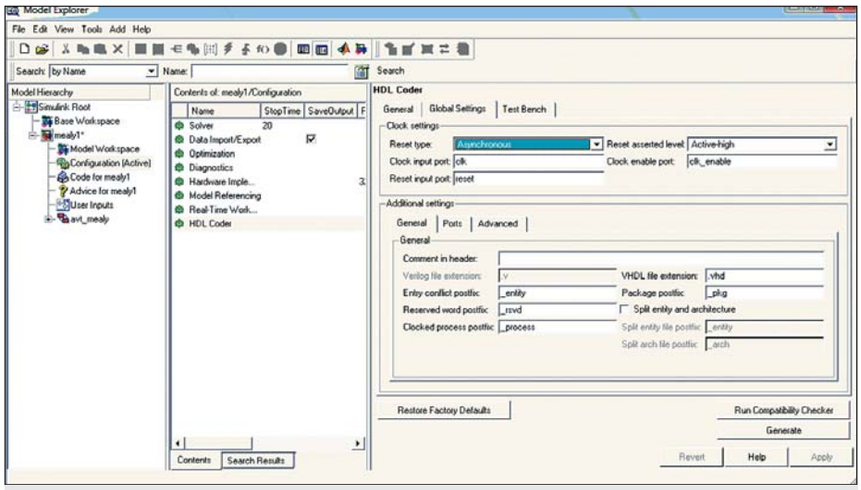

Рис. 2. Окно проводника модели. Настройка генератора кода языка VHDL

Автомат может принимать три состояния (рис. 1б): got_0, got_nickel, got_dime. Переходы по состояниям помечены цифрами. Когда состояние got_0 активно, возможны следующие переходы: брошена монета в 5 центов (Coin= =1), выход торгового автомата принимает значение Soda=0, а следующим активным состоянием будет got_nickel (переход 1). Если брошена монета в 10 центов (Coin= =2), то выход торгового автомата принимает значение Soda=0, а следующим активным состоянием будет got_dime (переход 2). Если не брошена ни одна из монет, то автомат остается в состоянии got_0. Остальные переходы по состояниям видны на рис. 1б.

После того как будет создана модель цифрового автомата, необходимо выбрать численный метод решения системы дифференциальных уравнений. С помощью проводника модели (Model Explorer) выбираем дискретный метод решения (discrete) в настройках Solver и настраиваем генератор кода языка VHDL в меню HDL Coder (рис. 2). Результат моделирования показан на рис. 1в, г.

Для получения кода на языке VHDL необходимо в проводнике модели нажать на кнопку Generate. При компиляции проекта цифрового автомата генератор кода языка VHDL, согласно ранее проведенным настройкам, автоматически добавляет сигнал тактирования clk, сигнал разрешения тактирования clk_enable, асинхронный сигнал сброса reset. Код автомата Мили на языке VHDL, полученный с использованием Simulink HDL Coder системы MATLAB/Simulink, показан в листинге 1. Здесь видно, что тип сигналов Coin и Soda — вещественный (real).

Анализируя стиль кодирования цифрового автомата, приходим к выводу, что метод кодирования не определен в коде языка VHDL.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
ENTITY avt_mealy IS
  PORT (
      clk : IN std_logic;
      clk_enable : IN std_logic;
      reset : IN std_logic;
      coin : IN real;
      soda : OUT real);
END avt_mealy;
ARCHITECTURE fsm_SFHDL OF avt_mealy IS
  TYPE T_state_type_is_avt_mealy is (IN_NO_ACTIVE_CHILD,
IN_got_0, IN_got_dime, IN_got_nickel);
  SIGNAL is_avt_mealy : T_state_type_is_avt_mealy;
  SIGNAL is_avt_mealy_next : T_state_type_is_avt_mealy;
BEGIN
  PROCESS (reset, clk)
        -- local variables
  BEGIN
        IF reset = '1' THEN
          is_avt_mealy <= IN_got_0;
        ELSIF clk'EVENT AND clk= '1' THEN
          IF clk_enable= '1' THEN
            is_avt_mealy <= is_avt_mealy_next;
          END IF;
        END IF;
    END PROCESS;
  avt_mealy : PROCESS (is_avt_mealy, coin)
    -- local variables
    VARIABLE is_avt_mealy_temp : T_state_type_is_avt_mealy;
  BEGIN
    is_avt_mealy_temp := is_avt_mealy;
    soda <= 0.0;
    CASE is_avt_mealy_temp IS
      WHEN IN_got_0 =>
          IF coin = 1.0 THEN
            soda <= 0.0;
            is_avt_mealy_temp := IN_got_nickel;
        ELSE
          IF coin = 2.0 THEN
            soda <= 0.0;
            is_avt_mealy_temp := IN_got_dime;
          END IF;
        END IF;
      WHEN IN_got_dime =>
          IF coin = 2.0 THEN
            soda <= 1.0;
            is_avt_mealy_temp := IN_got_nickel;
        ELSE
          IF coin = 1.0 THEN
            soda <= 1.0;
            is_avt_mealy_temp := IN_got_0;
          END IF;
        END IF;
      WHEN IN_got_nickel =>
          IF coin = 1.0 THEN
            soda <= 0.0;
            is_avt_mealy_temp := IN_got_dime;
        ELSE
          IF coin = 2.0 THEN
            soda <= 1.0;
            is_avt_mealy_temp := IN_got_0;
          END IF;
        END IF;
      WHEN OTHERS =>
        is_avt_mealy_temp := IN_got_0;
    END CASE;
      is_avt_mealy_next <= is_avt_mealy_temp;
  END PROCESS;
END fsm_SFHDL;
```

Листинг 1. Код автомата Мили на языке VHDL, полученный с использованием Simulink HDL Coder системы MATLAB/Simulink

Используется двухпроцессорный шаблон, оператор выбора CASE и перечисляемый тип данных (Enumerated type). Перечисляемый — это такой тип данных, при котором количество всех возможных состояний конечно. Его наиболее часто используют для обозначений состояний конечных автоматов. В этом случае есть возможность предоставить САПР ПЛИС использовать модуль логического синтеза и в зависимости от архитектуры ПЛИС самостоятельно выбирать метод кодирования [9]. Сигнал разрешения тактирования clk_enable генерируется как синхронный

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE ieee.std_logic_unsigned.all;
ENTITY avt_mealy IS
  PORT (
      clk : IN std_logic;
      clk_enable : IN std_logic;
      reset : IN std_logic;
      coin : IN std_logic_vector(1 downto 0);
      soda : OUT std_logic_vector (1 downto 0));
END avt_mealy;
ARCHITECTURE fsm_SFHDL OF avt_mealy IS
  TYPE T_state_type_is_avt_mealy is (IN_NO_ACTIVE_CHILD,
IN_got_0, IN_got_dime, IN_got_nickel);
  SIGNAL is_avt_mealy : T_state_type_is_avt_mealy;
  SIGNAL is_avt_mealy_next : T_state_type_is_avt_mealy;
BEGIN
  PROCESS (reset, clk)
      -- local variables
  BEGIN
      IF reset = '1' THEN
          is_avt_mealy <= IN_got_0;
      ELSIF clk'EVENT AND clk= '1' THEN
          IF clk_enable= '1' THEN
              is_avt_mealy <= is_avt_mealy_next;
          END IF;
      END IF;
  END PROCESS;
  avt_mealy : PROCESS (is_avt_mealy, coin)
      -- local variables
      VARIABLE is_avt_mealy_temp : T_state_type_is_avt_mealy;
  BEGIN
      is_avt_mealy_temp := is_avt_mealy;
      soda <= «00»;
      CASE is_avt_mealy_temp IS
          WHEN IN_got_0 =>
              IF coin = «01» THEN
                  soda <= «00»;
                  is_avt_mealy_temp := IN_got_nickel;
              ELSE
                  IF coin = «10» THEN
                      soda <= «00»;
                      is_avt_mealy_temp := IN_got_dime;
                  END IF;
              END IF;
          WHEN IN_got_dime =>
              IF coin = «10» THEN
                  soda <= «01»;
                  is_avt_mealy_temp := IN_got_nickel;
              ELSE
                  IF coin = «01» THEN
                      soda <= «01»;
                      is_avt_mealy_temp := IN_got_0;
                  END IF;
              END IF;
          WHEN IN_got_nickel =>
              IF coin = «01» THEN
                  soda <= «00»;
                  is_avt_mealy_temp := IN_got_dime;
              ELSE
                  IF coin = «10» THEN
                      soda <= «01»;
                      is_avt_mealy_temp := IN_got_0;
                  END IF;
              END IF;
          WHEN OTHERS =>
              is_avt_mealy_temp := IN_got_0;
      END CASE;
      is_avt_mealy_next <= is_avt_mealy_temp;
  END PROCESS;
END fsm_SFHDL;
```

Листинг 2. Код автомата Мили на языке VHDL в САПР ПЛИС Quartus

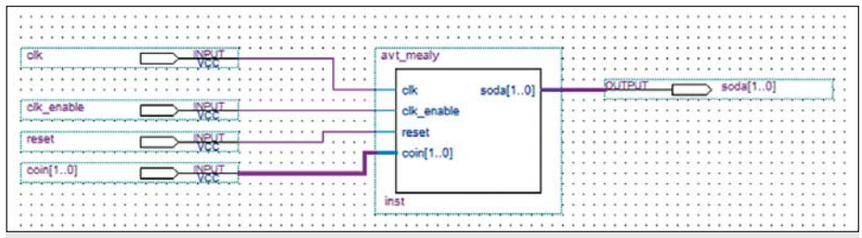

Рис. 3. Тестовая схема автомата Мили в САПР Quartus

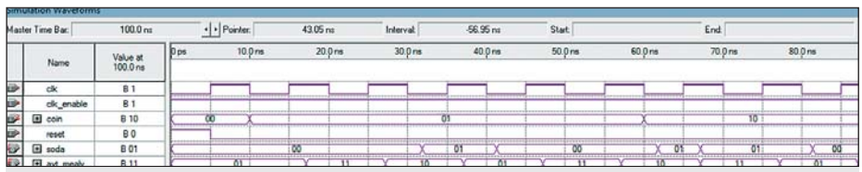

Рис. 4. Временная диаграмма автомата Мили в САПР Quartus. Тестируется переход по состояниям got_0, got_nickel, got_dime, got_0

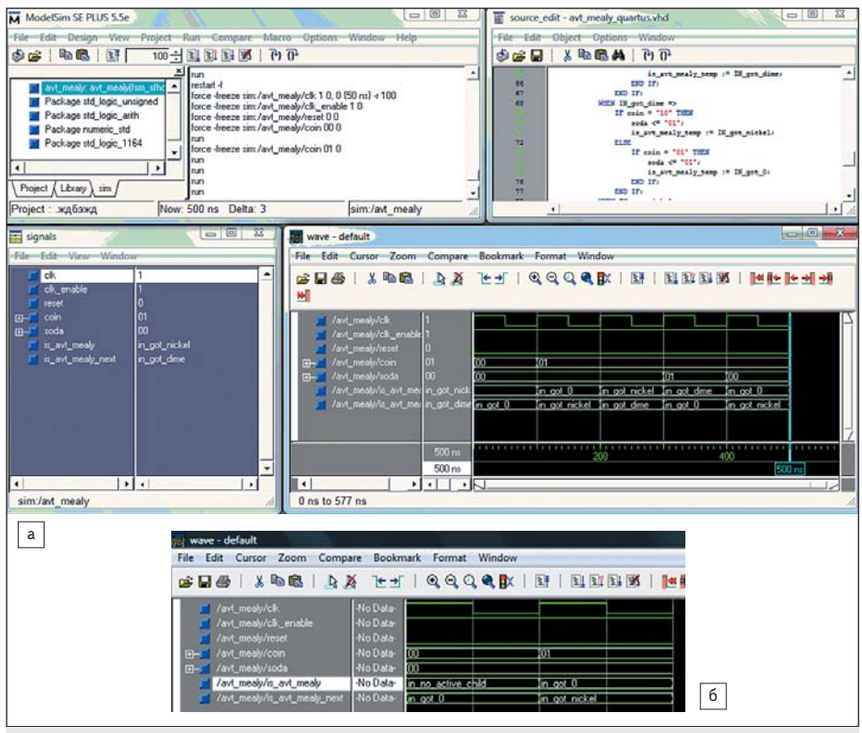

Рис. 5. Проектирование автомата Мили в ModelSim SE Plus: а) тестируется переход по состояниям got_0, got_nickel, got_dime, got_0; б) особенности отображения состояний автомата в векторном редакторе

(стоит после атрибута срабатывания по переднему фронту clk'EVENT AND clk= '1').

Код автомата Мили на языке VHDL, полученный с использованием Simulink HDL Coder системы MATLAB/Simulink, в САПР ПЛИС Quartus непосредственно использовать нельзя. Возникают ошибки компиляции. Для реализации проекта на базе ПЛИС фирмы Altera необходимо аналоговый сигнал Coin кодировать 2-битным цифровым сигналом Coin[1..0], действительным для одного такта сигнала Clk, показывающего монету, которую опустили:

- Coin[00] — B00 — нет монеты;
- Coin[01] — B01 — брошена монета в 5 центов (nickel);
- Coin[10] — B10 — брошена монета в 10 центов (dime)).

Сигнал Soda закодируем двухбитным сигналом Soda[1..0]:

- Soda[00] — нет бутылки;
- Soda[01] — бутылка.

«Подправленный» код автомата Мили на языке VHDL в САПР ПЛИС Quartus показан в листинге 2. Simulink HDL Coder кодировал переходы по состояниям в такой последовательности (листинг 1): вначале рассматривались все возможные переходы из состояния IN_got_0, затем из состояния IN_got_dime и IN_got_nickel. Поэтому состояния проектируемого автомата Мили в САПР ПЛИС Quartus кодируются в этой же последовательности: B01, B10, B11, то есть 1, 2, 3. На рис. 3 показана тестовая схема автомата Мили, а на рис. 4 — временная диаграмма. В процессе работы автомат «пробегает» по состояниям (сигнал/узел is_avt_mealy представляет собой регистр состояния, построенный на двухразрядной шине; на временной диаграмме узел

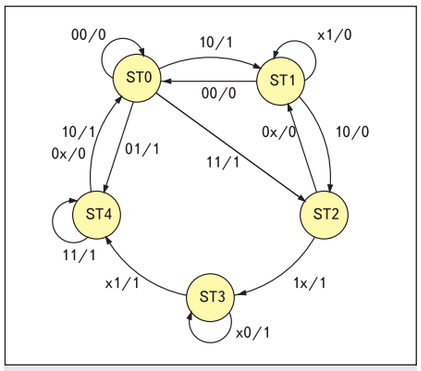

Рис. 6. Граф переходов автомата Мили

отображается значком контактной ножки с буквой R) с номерами B01, B11, B10 и B01, то есть 1, 3, 2, 1. Таким образом, тестируется переход по состояниям got_0, got_nickel, got_dime, got_0. Анализируя временную диаграмму, можно сделать вывод, что торговый автомат работает корректно. Если на вход будет подан сигнал Coin[11] — B11, то автомат по состояниям переходить не будет.

На рис. 5а показано проектирование автомата Мили в ModelSim SE Plus. В этом случае код языка не требует дополнительной «переделки». Также показаны состояния автомата, реализованные на сигналах is_avt_mealy (регистр текущего состояния) и is_avt_mealy_next (регистр следующего состояния). На рис. 5б видно, как кодируются состояния автомата. Состояние IN_NO_ACTIVE_CHILD является состоянием по умолчанию, введенным генератором кода Simulink HDL Coder. Компилятор САПР Quartus сокращает его на этапе компиляции проекта (минимизирует), а симулятор ModelSim SE Plus начинает работу именно с этого состояния. Сравнивая рис. 4 и 5, видим, что цифровые автоматы, спроектированные на различных платформах, работают корректно, несмотря на разные способы представления результатов моделирования. Однако в САПР ПЛИС Quartus сигнал soda B01 появляется асинхронно, а в ModelSim — с приходом тактового импульса, то есть синхронно.

Рассмотрим пример проектирования более сложного автомата Мили в «ручном» режиме. На рис. 6 показан автомат Мили, на рис. 7 тестовая схема, а на рис. 8 — временные диаграммы работы. В листинге 3 показан код языка VHDL по рекомендациям фирмы Actel [10]. Применяется классический вариант описания цифрового автомата — трехпроцессорный шаблон. Первый оператор процесса используется для описания блока регистров, как и в листингах 1–3, второй для описания логики переходов, а третий — для описания логики формирования выхода.

Таким образом, автоматически сгенерированный и оптимизированный код языка VHDL по графу переходов цифрового автомата, с использованием Simulink HDL Coder системы MATLAB/Simulink, позволяет значительно ускорить процесс разработки цифровых устройств для реализации их в базисе БИС и ПЛИС.

Литература

1. Stateflow and Stateflow Coder 7. Users Guide. www.mathworks.com.
2. Рабовалюк А. Обзор маршрута проектирования ПЛИС FPGA Advantage компании Mentor Graphics // Компоненты и технологии. 2005. № 9.
3. Рабовалюк А. Обзор маршрута проектирования ПЛИС FPGA Advantage компании Mentor Graphics // Компоненты и технологии. 2007. № 9.
4. Стешенко В. Б. Школа схемотехнического проектирования устройств обработки сигналов // Компоненты и технологии. 2000. № 3–6.
5. Стешенко В. Б. Школа схемотехнического проектирования устройств обработки сигналов. Занятие 8. Средства визуальной разработки цифровых автоматов // Компоненты и технологии. 2001. № 2.
6. Стешенко В. Примеры проектирования цифровых устройств с использованием языков описания аппаратуры // Схемотехника. 2001. № 7–9. www.platan.ru.
7. Стешенко В. Б. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры. М.: Издательский дом «Додэка-XXI», 2002.
8. Долинский М. Горячие темы EDA-индустрии // Компоненты и технологии. 2004. № 6.
9. Строгонов А. Проектирование конечных автоматов по методу ОНЕ // Компоненты и технологии. 2007. № 10.
10. Actel Digital Library. Q3 2001. Designing State Machines for FPGAs. September 1997. 97s05d18.pdf.

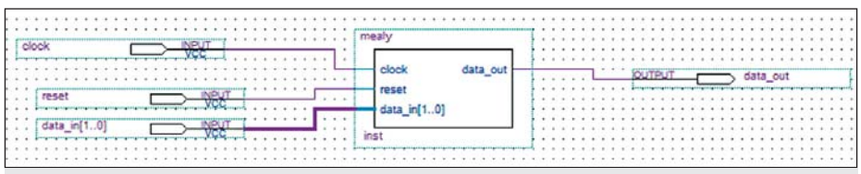

Рис. 7. Тестовая схема автомата Мили в САПР Quartus

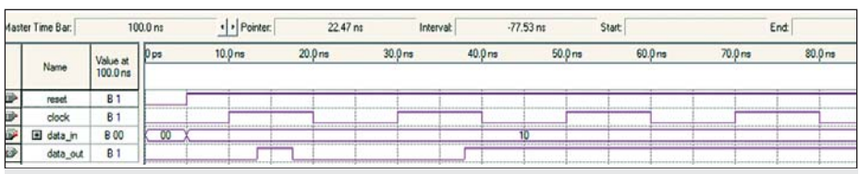

Рис. 8. Временная диаграмма работы проектируемого автомата

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY mealy IS
   PORT(clock, reset : IN STD_LOGIC;
        data_in : IN STD_LOGIC_VECTOR(1 downto 0);
        data_out : OUT STD_LOGIC);
END mealy;
ARCHITECTURE a OF mealy IS
TYPE state_values IS (st0, st1, st2, st3, st4);
signal state, next_state: state_values;
BEGIN
-- блок регистров
statereg: process(clock,reset)
begin
   if (reset = '0') then state<=st0;
      elsif (clock'event and clock='1') then
               state<=next_state;
   end if;
end process statereg;
-- комбинаторный блок (логика переходов)
process(state, data_in)
begin
case state is
when st0=>
case data_in is
when «00»=>next_state<=st0;
when «01»=>next_state<=st4;
when «10»=>next_state<=st1;
when «11»=>next_state<=st2;
when others => next_state<=st0;
end case;
when st1=>
case data_in is
when «00»=>next_state<=st0;
when «10»=>next_state<=st2;
when others => next_state<= st1;
end case;
when st2=>
case data_in is
when «00»=>next_state<=st1;
when «01»=>next_state<=st1;
when «10»=>next_state<=st3;
when «11»=>next_state<=st3;
when others => next_state<= st2;
end case;
when st3=>
case data_in is
when «01»=>next_state<=st4;
when «11»=>next_state<=st4;
when others => next_state<=st3;
end case;
when st4=>
case data_in is
when «11»=>next_state<=st4;
when others => next_state<=st0;
end case;
when others => next_state<=st0;
end case;
end process;
-- логика формирования выхода
process (state, data_in)
begin
case state is
when st0=>
case data_in is
when «00»=>data_out<='0';
when others => data_out<='1';
end case;
when st1=>data_out<='0';
when st2=>
case data_in is
when «00»=>data_out<='0';
when «01»=>data_out<='0';
when others => data_out<='1';
end case;
when st3=>data_out<='1';
when st4 =>
case data_in is
when «10»=>data_out<='1';
when «11»=>data_out<='1';
when others => data_out<='0';
end case;
when others => data_out<='0';
end case; end process; END a;
```

Листинг 3. Код автомата Мили на языке VHDL