Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Часть 3. Первая программа для микроконтроллера

Дмитрий ИОФФЕ
support@actel.ru
Андрей МАКСИМОВ
maksimov@actel.ru

Это третья статья из цикла, посвященного применению микропроцессорного IP-ядра 8051s для ПЛИС фирмы Microsemi. В [1] было рассмотрено построение аппаратной части системы на основе 8051s с использованием IP-ядер, поставляемых в составе САПР Libero. В [2] приведено описание ядра 8051s для программиста. Теперь мы попробуем создать простейшую программу для 8051s и запустить ее.

Введение

Для того чтобы в нашем устройстве заработала первая программа, мы должны сделать следующее:

- написать программу;
- откомпилировать ее;
- поместить ее в программную память микроконтроллера;
- провести моделирование работы программы;
- откомпилировать проект ПЛИС;
- «прошить» ПЛИС и проверить работу программы «в железе».

Приступим.

Пишем программу

Для начала определимся с инструментом. Для платформы 8051 существует много компиляторов разной степени доступности. Мы воспользуемся интегрированной средой разработки ПО для микропроцессоров SoftConsole корпорации Microsemi. Это бесплатный пакет, он входит в состав САПР Libero любой версии. В него встроен уже упоминавшийся компилятор SDCC.

Запустим SoftConsole. Отметим, что ее группа находится в меню **Пуск** отдельно от группы Libero. Сразу после первого запуска в окне **Workspace Launcher** нам предложат задать Workspace — рабочее пространство (рис. 1). Так называется папка, где будут храниться рабочие файлы SoftConsole. В принципе под рабочее пространство можно использовать любую папку. Но среда Libero в своем рабочем каталоге уже создала папку *firmware*. Удобно хранить в одном месте все файлы, относящиеся к проекту, поэтому укажем в окне **Workspace Launcher** именно эту папку.

Впоследствии можно будет выбрать другое рабочее пространство через меню **File → Switch → Workspace → Other**.

После определения рабочего пространства другие вопросы не будут заданы, и вскоре появится окно **SoftConsole**.

Теперь создадим новый проект. Из меню **File** пройдем по пунктам **New → C project**. Появится окно создания нового проекта (рис. 2).

Флажок **Use default location** оставляем установленным, чтобы новые файлы проекта попадали в заданное нами рабочее пространство. Выбираем в поле **Project Type** тип проекта Executable (Managed Make) — Empty Project и в поле **Toolchains** — строку Microsemi Core8051s Tools. Придумываем нашему проекту имя, например TestLED, вводим его в поле **Project Name** и нажимаем кнопку **Next**. Появится окно **Select Configuration** (рис. 3), где нужно будет выбрать конфигурацию проекта.

Здесь мы снимем флажок **Debug**, так как для нашей маленькой пробной программы отладка не понадобится, и нажмем кнопку **Advanced Settings**. Появится окно установ-

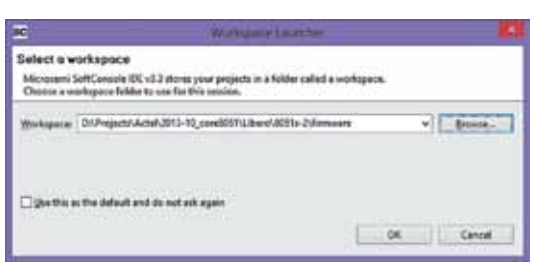

Рис. 1. Окно выбора рабочего пространства

Рис. 2. Создание нового проекта

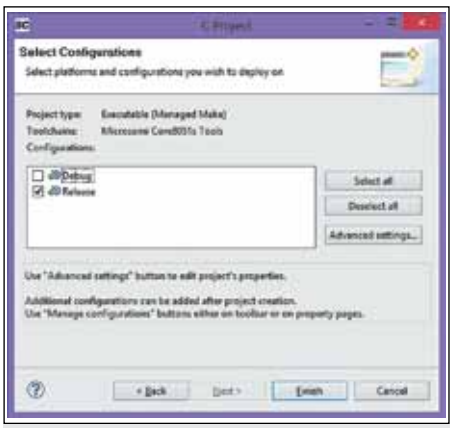

Рис. 3. Окно выбора конфигурации проекта SoftConsole

ки свойств нашего проекта **Properties for TestLED** (рис. 4).

Найдем в левом поле этого окна строку **C/C++ Build** и наведем на нее указатель мыши. Слева от строки появится маленький треугольник. Щелкнем по нему левой кнопкой мыши и выберем из развернувшегося списка пункт **Settings**. В ответ нам будет предложено выбрать опции для компилятора и компоновщика (linker). Щелкнем в группе опций компилятора **SDCC Compiler** по строке **Memory Options**, и в правом поле появится группа настроек памяти (рис. 5).

Выберем из выпадающего списка **Memory Model** пункт **Small** (**--model-small**). Тем самым мы зададим малую модель памяти, то есть переменные, для которых явно не указана область хранения, будут располагаться во внутреннем ОЗУ процессора. Это необходимо, так как в нашем учебном проекте внешнего ОЗУ просто нет.

Затем найдем такую же строку **Memory Options** в группе опций компоновщика **SDCC Linker** и аналогично зададим малую модель памяти. После этого нажмем кнопку **OK** и вернемся в окно **Select Configuration**, а в нем нажмем на кнопку **Finish**.

Мы снова работаем с главным окном **SoftConsole**. Теперь в левой части окна

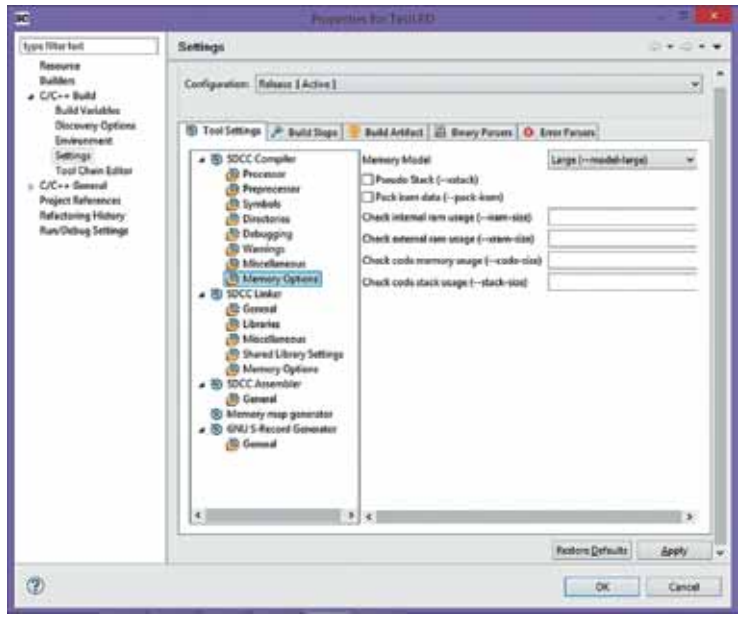

Рис. 5. Установка опций памяти

SoftConsole, в окне **Project Explorer**, которое предназначено для навигации по проекту, появился значок папки с именем проекта.

Прежде чем писать программу, нам необходимо запастись заголовочными файлами. Файл со смещениями адресов устройств шины APB уже есть: в среде Libero он расположен в упомянутой папке firmware. Его имя состоит из имени проекта и строки _hw_platform, таким образом, для нашего проекта получилось *System8051s_hw_platform.h*. Осталось получить файл с определением регистров процессора *reg51.h*.

Самый простой способ создать этот файл — скопировать через буфер обмена из руководства [3]. Однако во избежание возможных ошибок лучше затратить немного времени и загрузить с сайта корпорации Microsemi драйвер аппаратного уровня абстракции (HAL). Среди загруженных файлов будет и *reg51.h*. Вообще в рамках работы над нашими учебными проектами мы не будем использовать фирменные драйверы устройств. Это позволит нам сэкономить дорогостоящую память программ, которую мы будем создавать без использования блочной памяти. Но впоследствии драйверы могут нам пригодиться, и сейчас мы посмотрим, как это делается.

В состав Libero входит утилита Firmware Catalog. Ярлык для ее вызова есть в главном меню Windows в группе Libero. Запустим ее.

Замечание (для тех, кто активно пользуется брандмауэрами (firewall)). Конечно, мы вправе запретить любой программе выход в Интернет. Но многие компоненты Libero и SoftConsole, как и других САПР, используют сетевые протоколы для взаимодействия между собой. Брандмауэр, конечно, контролирует и такие операции. Поэтому нужно внимательно читать его сообщения и разрешать обращения по локальным адресам, иначе ПО не будет нормально работать. Например, не будет работать справка

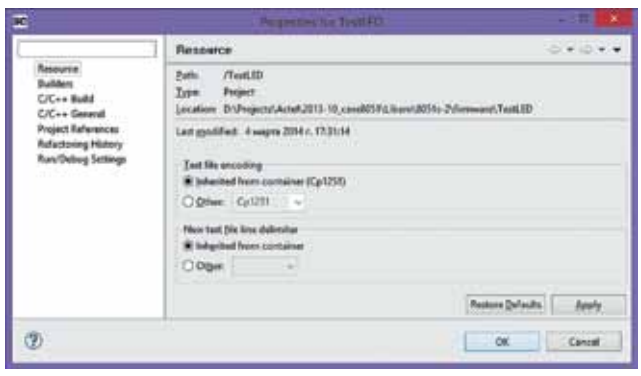

Рис. 4. Окно установки свойств проекта SoftConsole

Рис. 6. Загрузка драйвера HAL

SoftConsole. А программа Firmware Catalog просто не может обойтись без Интернета.

При первом запуске Firmware Catalog подключится к Интернету и скачает большой список доступных на данный момент драйверов. Введем в поле **Search** буквы HAL, и в списке останутся только два драйвера, среди которых мы легко найдем нужный (рис. 6).

Щелкнем мышью по строке **Hardware Abstraction Layer (HAL)**, а затем по кнопке **Download**. Драйвер будет загружен, и его название будет выделено жирным шрифтом. Теперь нужно выполнить «генерацию» драйвера, то есть скопировать файлы драйвера в наш рабочий каталог. Нажмем теперь на кнопку **Generate**. Появится окно **Generate Options** (рис. 7).

Рис. 7. Окно опций генерации драйвера HAL

Нажмем в этом окне кнопку с тремя точками и укажем на ту же папку *firmware*, а затем нажмем на кнопку **OK**. Появится окно конфигурации драйвера HAL (рис. 8).

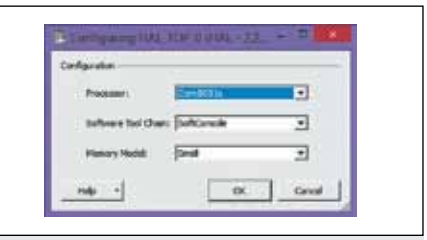

Рис. 8. Окно конфигурации драйвера HAL

Выберем из выпадающего списка в поле **Processor** строку **8051s**, а в полях **Software Tool Chain** и **Memory Model** оставим соответственно **SoftConsole** и **Small**. Нажмем **OK** и посмотрим на отчет о генерации драйвера (рис. 9).

Здесь мы видим, что в папке *firmware* появилась папка *HAL*, а в ней в каталоге ***\Core8051s\SDCC*** лежит нужный нам

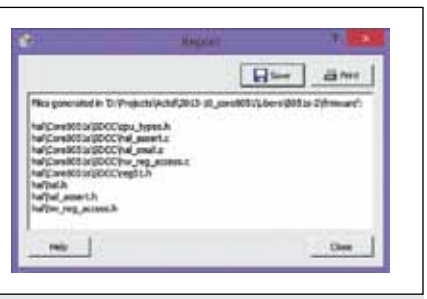

Рис. 9. Отчет о генерации драйвера

файл *reg51.h*. Закроем отчет, а затем и окно программы **Firmware Catalog**. Теперь можно переместить полученный файл в папку *firmware*, а папку *HAL* стереть, она нам в ближайшее время не понадобится.

Теперь переходим к тексту программы. Пусть для начала это будет совсем простая программа, которая будет отслеживать положение микропереключателя и в соответствии с ним зажигать или гасить светодиод. Доступ к микропереключателю и светодиоду осуществляется через IP-ядро CoreGPIO. Создадим исходный текст проекта на языке С. Начнем с того, что выберем из главного меню окна **SoftConsole** пункт **File** и проследуем далее по пунктам **New → Source File**. Появится окно **New Source File** (рис. 10).

Рис. 10. Окно New Source File

В поле **Source Folder** уже стоит выбранное нами имя проекта **TestLED**. Введем в поле **Source File** имя нового файла, обязательно с расширением — пусть это будет *MainTestLED.c*. Из выпадающего списка Template можно выбрать шаблон **Default C source template** или **None**. Нажимаем на кнопку **Finish**, и в центре окна **SoftConsole** появляется окно редактора с текстом нашей будущей программы.

Введем в это окно следующий текст:

```
#include "..\System8051s_hw_platform.h"
#include "..\reg51.h"

/*Обращение к шине APB:*/
#define APB_base 0xF000U //начало адресного пространства APB
#define GPIO_addr APB_base + COREGPIO_0 //адрес блока GPIO

/*Маски для обращения к линиям ввода/вывода:*/
#define jumper 0x01 //вход опроса переключателя
#define LED_1 0x02 //первый светодиод

__xdata at GPIO_addr + 0xA0 unsigned char dataout;
__xdata at GPIO_addr + 0x90 unsigned char datain;

unsigned char inreg;
unsigned char outreg;

void main ()
{
  outreg = 0x00;
  for (;;) {
    inreg = datain;
    if ((inreg & jumper) == 0)
      outreg = outreg & ~LED_1;
    else
      outreg = outreg | LED_1;
    dataout = outreg;
  }
}
```

Чтобы определить смещение портов ввода (0x90) и вывода (0xA0) модуля GPIO, следует заглянуть в описание IP-ядра [4].

Компиляция программы

Чтобы откомпилировать программу, нужно в **SoftConsole** вызвать пункт меню **Project → Build All** или нажать одновременно клавиши **Ctrl** и **B**. Если компиляция пройдет без ошибок, то в папке **Release** нашего проекта среди прочих файлов появится файл *TestLED.ihx*. Это и есть программа для нашего микроконтроллера в шестнадцатеричном коде (*HEX*-файл), которую мы должны загрузить в память программ.

Создание памяти программ

В первой части нашей статьи мы создали временную заглушку вместо настоящего ПЗУ, чтобы проверить проект. Это была простая комбинационная схема, и 8051s работать с такой памятью не будет. Поэтому мы должны создать синхронную схему, которая выдает коды на шину данных по спаду импульсов тактовой частоты, а также обнуляет выходной регистр по сигналу сброса. То есть VHDL-код должен будет выглядеть примерно так:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity MyROM is
port (
  clock: in std_logic; -- active falling edge
  n_reset: in std_logic; -- active low
  ReadEnable: in std_logic; --active high
  Addr: in std_logic_vector(15 downto 0);
  DataOut: out std_logic_vector(7 downto 0)
);
end MyROM;

architecture MyROM_0 of MyROM is

  constant AddrSize : natural := 9;
  signal MemData: std_logic_vector(7 downto 0);

  -- std_logic_vector to decimal number conversion:
  function SLVtoNatural (SLV: std_logic_vector) return natural is
    variable result : natural;
  begin
    result := 0;
    for i in 0 to SLV'length - 1 loop
      if SLV(i) = '1' then result := result + 2**i; end if;
    end loop;
    return result;
  end SLVtoNatural;

begin

ROM_EVAL: process(Addr)
  variable MyAddr: std_logic_vector(AddrSize-1 downto 0);
  variable Index: natural;
begin
  MyAddr := Addr(AddrSize-1 downto 0);
  Index := SLVtoNatural(MyAddr);
  case Index is
    when 0 => MemData <= x"02";
    when 1 => MemData <= x"00";
… здесь в операторе case будут закодированы все ячейки памяти
    when 242 => MemData <= x"22";
    when others => MemData <= "XXXXXXXX";
  end case;
end process ROM_EVAL;

DataOut_CHANGE: process (n_reset, clock, MemData)
begin
  if (n_reset = '0') then
    DataOut <= (others => '0');
  elsif (falling_edge(clock)) then
    DataOut <= MemData;
  end if;
end process DataOut_CHANGE;

end MyROM_0;
```

Рис. 11. Окно конвертера HEX-файла в коде на VHDL

Осталось выяснить, как перевести наш шестнадцатеричный код, полученный после компиляции программы, в такой текст на VHDL.

Формат *HEX*-файлов описан в [5]. Конечно же, вручную делать такое преобразование немыслимо. Но инженеру уместно владеть хоть одним языком программирования. Создание программы-конвертера в среде Delphi 2010 заняло около трех часов, включая поиск красивой иконки в Интернете. И теперь этот конвертер вместе с исходным кодом можно найти по адресу [6]. Отметим, что он может работать только с *HEX*-файлами *intel-standart*.

Запустим конвертер (рис. 11).

Порядок работы с конвертером следующий. Сначала мы нажимаем на кнопку **Select HEX File** и в открывшемся диалоговом окне указываем наш файл с расширением *.IHX*. Закрываем диалоговое окно. Конвертер может также преобразовывать файлы с расширением *.HEX*. Такой файл мы бы получили бы, если бы разрешили компилятору создать отладочную версию кода (Debug). Затем нажимаем на кнопку **Set VHDL File** и находим в открывшемся диалоговом окне папку *HDL* проекта Libero. Вводим имя файла VHDL, например MyROM, и закрываем диалоговое окно. Далее можно ограничить размер ПЗУ. Если наш *IHX*-файл не помещается в ПЗУ такого размера, то конвертер выдаст сообщение об ошибке. И наконец, нажимаем на кнопку **Convert**. Код модуля памяти на VHDL создан.

При выходе конвертер запоминает имена входного и выходного файлов и размер ПЗУ, и при следующих запусках их можно не указывать.

Теперь нам нужно вставить полученное ПЗУ в проект, который мы описали в первой части статьи. Открываем проект в Libero, находим старый модуль MyROM, щелкаем по нему правой кнопкой мыши и выбираем из выпавшего меню пункт **Update Instance(s) with Latest Component**. Старый MyROM заменяется на новый. Делаем следующие соединения:

- вход clock модуля MyROM соединяем с системной тактовой частотой SYSCLK;
- его же вход n_reset — с выходом PRESETN ядра 8051s.

Отметим и запомним: после каждого изменения программы и, соответственно, новой компиляции следует заново проводить генерацию проекта Libero.

И еще нам нужно исправить две ошибки, допущенные авторами в первой части статьи. Откроем окно конфигурации шины APB и внесем следующие изменения:

- Из выпадающего списка **Number of address bits driven by master** выберем 12, так как 8051s по-другому работать не умеет.
- Из списка **Position in slave address of upper 4 bits of master address** выберем [11:8].

И для тех, у кого такая же отладочная плата: в файле *.PDC* первая рабочая строка должна иметь вид set_io NSYSRESET-pinname T19.

Теперь проверяем проект: в меню **Libero SmartDesign** выбираем пункт **Check Rules Design**. Если ошибок нет, то из меню **SmartDesign** выбираем **Generate Design**. Через некоторое время мы получим сообщение об успешной генерации проекта.

Моделирование

Отлаживать нашу программу из нескольких строк в среде SoftConsole вряд ли полезно: весь текст хорошо виден, ошибки маловероятны. Гораздо интереснее моделировать наш проект в Modelsim, чтобы убедиться, что в маршруте проектирования нет ошибок и вся аппаратура, включая ПЗУ программ, работает правильно.

Во время генерации проекта был автоматически создан файл *testbench.vhd*. Он находится в папке нашего проекта, в подпапке ***\component\work\System8051s***. В нем на входы проекта подаются тактовая частота и сигнал сброса. Нам нужно проверить реакцию системы на движение микропереключателя. Для этого следует подать на вход GPIO_IN не логический ноль, как в исходном тестбенче, а некий нужный нам сигнал, назовем его jumper. Для предотвращения каких-либо неожиданностей создадим новый тестбенч и сохраним его в папке *stimulus* нашего проекта под другим именем, например *tb_inout.vhd*:

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_inout is
end tb_inout;

architecture behavioral of tb_inout is
   constant SYSCLK_PERIOD : time := 20.8333 ns;
   signal SYSCLK : std_logic := '0';
   signal NSYSRESET : std_logic := '0';
   signal Jumper : std_logic := '0';
   component System8051s
      port(
         -- Inputs
         NSYSRESET : in std_logic;
         SYSCLK : in std_logic;
         GPIO_IN : in std_logic;
         -- Outputs
         GPIO_OUT : out std_logic_vector(2 downto 1)
      );
   end component;

begin
   process
      variable vhdl_initial : BOOLEAN := TRUE;
   begin
      if ( vhdl_initial ) then
         NSYSRESET <= '0';
         wait for ( SYSCLK_PERIOD * 200 );
         NSYSRESET <= '1';
         wait;
      end if;
   end process;
   SYSCLK <= not SYSCLK after (SYSCLK_PERIOD / 2.0 );
   Jumper <= not Jumper after (SYSCLK_PERIOD * 1000 );
    System8051s_0 : System8051s
      port map(
         NSYSRESET => NSYSRESET,
         SYSCLK => SYSCLK,
         GPIO_IN => Jumper,
         GPIO_OUT => open
      );
end behavioral;
```

Теперь нужно сообщить среде Libero, что мы хотим работать именно с этим тестбенчем. Выберем в главном меню **Libero** пункт **Project → Settings** и в появившемся окне настроек проекта щелкнем по закладке **Simulation** (рис. 12).

Рис. 12. Установка параметров моделирования

Выберем в группе ModelSim Options пункт DO File, если этого еще не сделано. Установим в поле **Simulation Runtime** время моделирования 50 мкс (50us) и зададим в качестве входного наш файл *tb_inout*. Больше здесь пока ничего изменять не будем и нажмем кнопку **OK**.

Затем в поле **Design Explorer** главного окна **Libero** щелкнем правой кнопкой мыши по имени проекта System8051s и выберем из выпавшего меню пункт **Organize Stimulus**. Откроется окно управления тестбенчами (рис. 13).

Щелкнем левой кнопкой мыши по имени файла *testbench.vhd* в правом поле окна и удалим его оттуда, нажав на кнопку Remove. Затем прокрутим список в левой части окна

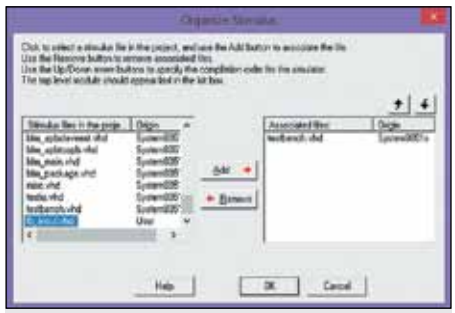

Рис. 13. Окно Organize Stimulus

и найдем там наш файл *tb_inout.vhd*, щелкнем по нему мышью и нажмем на кнопку **Add**. Закроем окно, нажав на кнопку **OK**.

Теперь можно начинать моделирование. Щелкнем правой кнопкой мыши по названию проекта в поле **Design Explorer** среды Libero и из выпавшего меню выберем пункт **Run Pre-Synthesis Simulation**. Запустится программа Modelsim, и через некоторое время, если не обнаружатся какие-либо ошибки, мы увидим ее окно с временной диаграммой, на которой отобразятся три сигнала, описанные в файле *tb_inout.vhd*.

Работу с системой Modelsim мы здесь рассматривать не будем, это тема отдельной статьи или даже цикла. Опишем здесь только несколько действий, которые потребуется выполнить на начальном этапе, в рамках работы с нашим проектом.

Добавим в задание для моделирования выходные сигналы проекта. В левой части окна Modelsim находится список различных элементов (Instances), имеющихся в проекте (рис. 14).

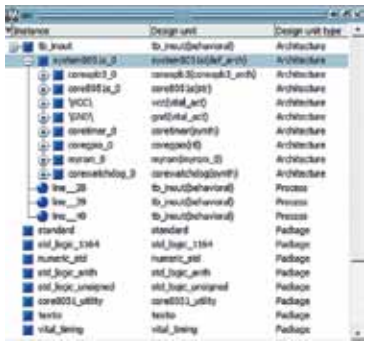

Рис. 14. Список Instances

Щелкнем мышью по имени проекта System8051s_0. В средней части окна Modelsim отобразится список объектов верхнего уровня, поведение которых мы можем моделировать (рис. 15). Отметим, что, перемещаясь по дереву проекта, мы можем выбирать для моделирования сигналы из вложенных модулей, например MyROM или CoreGPIO. Правда, их поведение можно увидеть только при моделировании до синтеза.

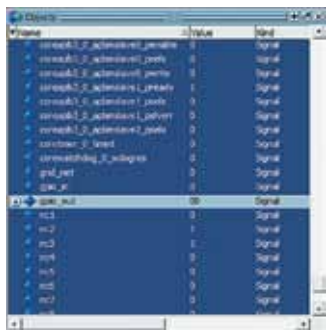

Рис. 15. Список объектов для моделирования

Найдем в этом списке выходы gpio_out и перетащим их мышью в правую часть окна Modelsim, в область отображения моделируемых сигналов Wave. И сразу, чтобы не забыть об этом второпях, сохраним полученный вид окна моделируемых сигналов для будущих сеансов работы. Он будет сохраняться в файле с расширением *.do*. Выберем из главного меню Modelsim пункт **File → Save Format** (он отображается только тогда, когда активна область Wave). По умолчанию Modelsim предлагает сохранить файл в папке Simulation нашего проекта. Это не очень удачный вариант, так как в случае непонятного поведения Modelsim, когда все вроде бы правильно, а модель не работает, обычно эту папку стирают со всем ее содержимым. Лучше сохранить файл прямо в папке проекта, под индивидуальным именем, например *mywave.do* (обязательно с расширением). Теперь перейдем к Libero и снова вызовем окно настроек проекта через меню **Project → Settings**, откроем закладку **Simulation** и в поле **ModelSim Options** выберем строку **Waveforms** (рис. 16).

Установим флажок **Include DO File**, затем в поле **Included DO File** нажмем на кнопку с тремя точками и в открывшемся диалоговом окне укажем на наш файл *mywave.do*. Теперь нажмем на кнопку **OK** и закроем окно. При следующих запусках Modelsim из Libero мы увидим заданные нами временные диаграммы.

Рис. 16. Определение файла временных диаграмм для моделирования

Продолжим работу с Modelsim. Теперь нам нужно увидеть, как ведут себя выходы gpio_out.

В верхней части окна Modelsim есть панель **Simulate** (рис. 17). Заменим в поле интервала моделирования установленное по умолчанию значение 100 пс на нужное нам, например 50 мкс. Затем щелкнем по кнопке **Restart**. (Она находится слева от этого поля, ее название появится в желтом ярлычке, если навести на кнопку указатель мыши.) Появится небольшое окно **Restart**. Сейчас мы не будем анализировать его содержание и просто нажмем кнопку **OK**. Затем нажмем на кнопку **Run**, она находится справа от поля с интервалом моделирования. После этого моделирование начнется сначала, и теперь, если все правильно, мы увидим переключение выхода GPIO_OUT [1] (рис. 18).

Если нам нужно изменить участок отображения временной диаграммы, мы можем воспользоваться кнопками панели **Zoom** (рис. 19). Для просмотра части отображаемой временной диаграммы следует сделать в пределах этой части движение мышью слева направо (или справа налево) и вниз при нажатой средней кнопке.

Рис. 17. Панель Simulate

Рис. 19. Панель Zoom

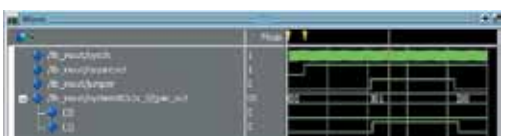

Рис. 18. Первый результат моделирования

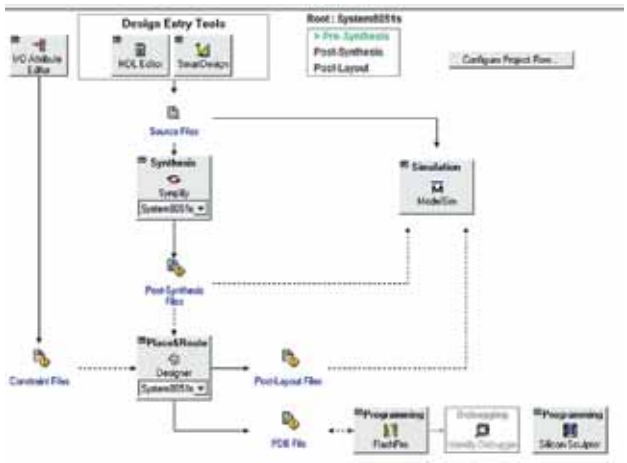

Рис. 20. Управление проектированием

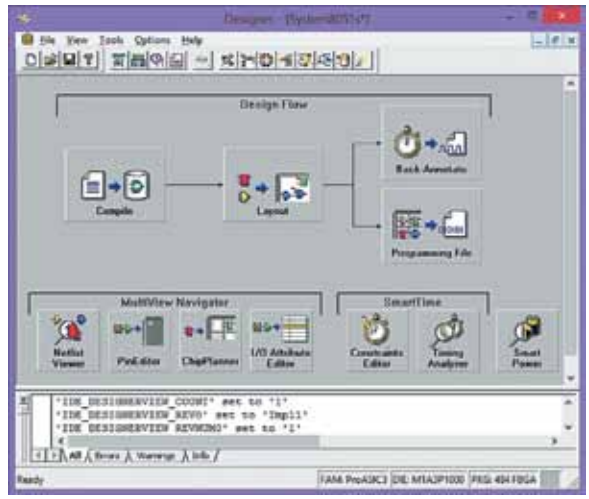

Рис. 21. Окно Designer

Компиляция проекта и шаг назад

Если моделирование прошло успешно, мы можем приступить к компиляции проекта. Закроем Modelsim и вернемся к Libero. В центральной части окна переключимся на закладку **Project Flow** (рис. 20).

Нажмем на большую кнопку **Synthesis**. Тем самым мы запустим синтезатор Synplify. В появившемся окне **Organize Constraints for Syntesis** нажмем на **OK**, не вдаваясь пока в подробности. Запустится синтезатор. Рассматривать его в этой статье мы не будем: это тема отдельной статьи, как и Modelsim. Просто найдем в его левом верхнем углу большую кнопку **Run** и нажмем на нее. Название кнопки сменится на **Cancel** и, через какое-то время, снова на **Run**. Синтез окончен.

Очень полезно просмотреть список сообщений синтезатора на закладке **Messages** в нижней части его окна. Там много предупреждений. Например, синтезатор сообщит, если списки чувствительности процессов VHDL не соответствуют реальному составу входных сигналов процесса. Это может привести к несовпадению результатов моделирования до и после синтеза.

Закроем окно синтезатора. Вернувшись в Libero, мы увидим, что кнопка **Synthesis** в **Project Flow** стала зеленой. Значит, теперь можно нажимать на кнопку **Place&Route**. Сделаем это. Запустится компонент среды Libero — Designer. В небольшом окне **Audit Status** нам предложат выбор по умолчанию Re-Import All Source Files. Согласимся с этим выбором и нажмем на кнопку **OK**, а в следующих двух окнах тоже нажмем **OK**. После этого мы увидим окно **Designer** (рис. 21).

Нажмем на большую кнопку **Compile**, согласимся с предложением в небольшом окне (и дальше будем поступать аналогично, пока не научимся предлагать что-то лучшее) и подождем, пока кнопка **Compile** станет зеленой. Теперь нажмем на кнопку **Layout**, согласимся с предложенными **Layout Options** и дождемся окончания трассировки. Ждать придется некоторое время. Когда кнопка **Layout**, в свою очередь, станет зеленой, это будет означать, что мы можем приступать к генерации файла для программатора.

Однако здесь нас подстерегает неприятный сюрприз. Все это время мы не задумывались о реальном быстродействии нашей ПЛИС. Среда Libero обошлась без наших указаний на этот счет, но отчет ее временно́го анализатора обязателен для просмотра. Нажмем в окне **Designer** на кнопку **Timing Analyzer**. Появится большое окно, в котором нам нужно найти таблицу Clock Details (рис. 22).

Из этой таблицы следует, что наш проект может работать на тактовой частоте до 22,372 МГц. А тактовый генератор у нас на отладочной плате работает на частоте 48 МГц, что в два с лишним раза больше. Можно было обнаружить это и раньше, еще работая над тем, что описано в первой части статьи. Но там при другом стечении обстоятельств мы могли бы случайно получить даже небольшой запас по частоте, который мог исчезнуть после компиляции проекта с окончательной версией памяти программ.

Рис. 22. Таблица Clock Details из отчета временного анализатора

Самый простой выход из положения в нашем случае — поделить входную тактовую частоту на четыре и тактировать нашу систему результатом деления. Использовать в качестве делителя модуль Counter из каталога Libero не получится: ему нужен сброс, а сигнал сброса у нас один, при этом остальная часть проекта должна получать свой сигнал сброса при уже работающей тактовой частоте. Поэтому напишем на языке описания аппаратуры небольшой модуль и назовем его, например, clock_start:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity clock_start is
   port (
      clock: in std_logic;
      n_reset: in std_logic;
      outclock: out std_logic;
      outnreset: out std_logic
   );
end clock_start;

architecture clock_start_0 of clock_start is
   constant module: natural := 4;
   signal divider: natural range 0 to module - 1;
   constant reset_time: natural := 40;
   signal reset_counter: natural range 0 to reset_time - 1;
begin
outclock_l:
   process(clock, n_reset)
   begin
      if n_reset = '0' then divider <= 0;
      elsif rising_edge(clock) then
         if divider = module - 1 then divider <= 0;
         else divider <= divider + 1; end if;
         if divider < 2 then outclock <= '0'; else outclock <= '1'; end if;
      end if;
   end process outclock_l;
outnreset_l:
   process(clock, n_reset)
   begin
      if n_reset = '0' then reset_counter <= 0;
      elsif rising_edge(clock) then
         if reset_counter /= reset_time - 1 then reset_counter <= reset_
counter + 1; end if;
         if reset_counter = reset_time - 1 then outnreset <= '1'; else
outnreset <= '0'; end if;
      end if;
   end process outnreset_l;
end clock_start_0;
```

Этот модуль нужно положить в папку *hdl* нашего проекта. После этого он появится в поле **Design Explorer**. Щелкнем

по нему правой кнопкой мыши и выберем из выпавшего меню пункт **Instantiate in System8051s**. После этого новый модуль появится на холсте. Отметим, что он может появиться в неожиданном месте, например, за границами видимой части экрана. В этом случае нужно разыскать на панели инструментов Libero кнопку **Zoom Fit** (по всплывающей желтой подсказке) и нажать на нее. После этого мы увидим весь экран, и можно будет перетащить новый модуль поближе к остальной части проекта и вернуть прежний масштаб.

Теперь подключим clock_start к нашему проекту. Найдем на краю холста порт SYSCLK, щелкнем правой кнопкой мыши по цепи, соединяющей его со схемой (она должна при этом изменить цвет), и выберем из выпавшего меню пункт **Delete Net**. Затем соединим освободившийся порт с входом clock модуля clock_start, а выход outclock этого модуля — с тактовыми входами всех остальных компонентов проекта. Соединения делаем так, как описано в [1]. После этого аналогично разорвем связь между портом NSYSRESET и одноименным входом ядра 8051s, затем соединим этот вход с выходом outnreset модуля clockstart, а вход n_reset этого модуля — с портом NSYSRESET.

Далее проверяем наш проект на отсутствие ошибок и снова выполняем полную компиляцию. Полезно повторить моделирование, увеличив его время в четыре раза. Потом в окне **Designer** нажимаем кнопку **Programming File**. Дожидаемся, пока эта кнопка станет зеленой, и закрываем окно **Designer**. Файл с расширением *.pdb* для программатора получен.

Программирование ПЛИС и проверка работы

Настало время подключить программатор (рис. 23) и отправить нашу прошивку в ПЛИС. На отладочных платах программатор встроен в саму плату, поэтому последнюю следует просто подключить кабелем к USB-порту компьютера. Для обычных схем нужно использовать автономный программатор Silicon Sculptor или внутрисхемный программатор FlashPro. Относительно программирующего модуля среды Libero встроенный программатор отладочной платы ничем не отличается от FlashPro.

Подключаем нашу отладочную плату к компьютеру и подаем на нее питание, затем нажимаем левую кнопку **Programming** в поле **Project Flow** — ту, на которой написано **Flash Pro**. Откроется окно **FlashPro**.

На встроенном программаторе некоторое время будет мигать желтый светодиод, как произошло бы и с Flash Pro. По окончании программирования снимем питание с платы и вновь подадим его. Может быть, в нашем случае это и не нужно, но привыкнуть к этому стоит.

Теперь возьмем подходящий тонкий предмет и начнем изменять положение микропереключателя. Мы увидим, как зажигается и гаснет наш светодиод. Конечно, для этого есть более простые способы, но зато как многому мы научились!

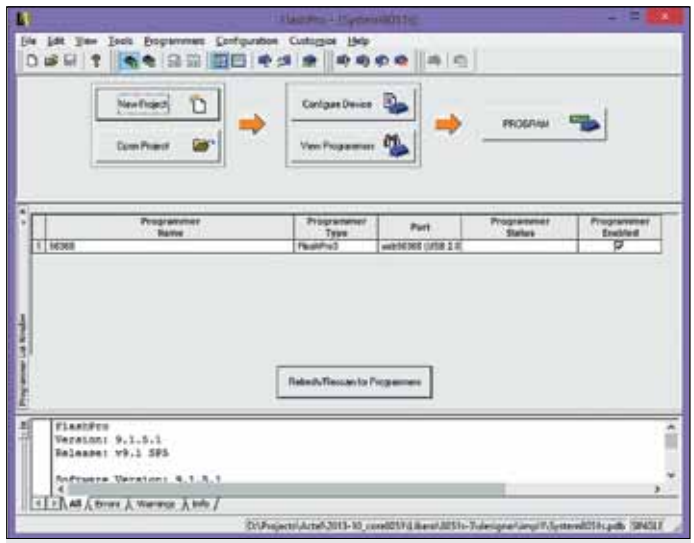

Рис. 23. Окно управления программатором

Выводы

В третьей части статьи мы написали простейшую программу для микропроцессорного ядра 8051s корпорации Microsemi и проверили ее работу на реальной плате. Предлагаемый порядок действий при работе с программой получился такой:

- редактируем текст программы;
- компилируем его, получая файл *.IHX* (*.HEX*);
- обновляем модуль памяти, созданный на языке VHDL, при помощи специального конвертера;
- генерируем проект Libero заново.

После этого можно моделировать проект Libero или компилировать его для получения файла прошивки *.PDB*. Файлы учебного проекта можно найти в Интернете по адресу [7].

В следующей части статьи мы расширим наш проект: добавим в него работу по прерываниям и задействуем сторожевой таймер.

Литература

1. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi // Компоненты и технологии. 2014. № 1.
2. Иоффе Д., Максимов А. Разработка проекта микроконтроллера 8051s на основе IP-ядер корпорации Microsemi. Ч. 2. IP-ядро 8051s для программиста // Компоненты и технологии. 2014. № 3.
3. Core8051s v2.4 Handbook. www.microsemi.com
4. CoreGPIO v3.0 Handbook. www.microsemi.com
5. Intel HEX. http://ru.wikipedia.org/wiki/Intel_HEX
6. Конвертер HEX-файла в код на VHDL. http://www.actel.ru/catalog/HexToVHDL.zip
7. http://www.actel.ru/catalog/8051s-2.zip