МИКРОКОНТРОЛЛЕРЫ КОМПАНИИ "МИЛАНДР" – ЭФФЕКТИВНОЕ СРЕДСТВО ПРОГРАММИРОВАНИЯ ПЛИС

В.Комиссаров spezialist@list.ru

Сегодня очень популярны микроконтроллеры на базе ядра ARM Cortex-M. Все ведущие производители элементной базы электроники (STMicroelectronics, Texas Instruments, Atmel, NXP и др.) имеют в ассортименте представителей этого семейства. В России микроконтроллеры с ядром Cortex-M выпускает компания "Миландр". Их можно использовать, в частности, для внутрисхемного программирования ПЛИС. О том, как наиболее эффективно организовать программирование ПЛИС фирмы Xilinx с помощью микроконтроллеров компании "Миландр" рассказывается в предлагаемой статье.

Реализацию программирования ПЛИС рассмотрим на примере микроконтроллера MDR32F9Q2I [1]. Он имеет 32-битное ядро ARM Cortex-M3, его тактовая частота может достигать 80 МГц. Микроконтроллер оснащен 128 Кбайт энергонезависимой памяти и встроенным ОЗУ размером 32 Кбайт, питается от источников напряжением 2,2–3,6 В, может работать как от внешнего кварцевого, так и от внутреннего генераторов. MDR32F9Q2I имеет несколько режимов пониженного энергопотребления, стандартный набор интерфейсов (UART, SPI, I^2C, CAN), три таймера и контроллер DMA. Особенность микроконтроллера – наличие модуля USB OTG, что дает возможность применять его как в качестве ведущего, так и ведомого устройства.

Есть несколько способов программирования ПЛИС Xilinx с помощью микроконтроллера MDR32F9Q2I. Один из них – использование так называемого LoonBoard Unified Bootloader (LUB) [2]. Это загрузчик, позволяющий работать как с бинарным файлом конфигурации ПЛИС, так и с файлом прошивки для контроллеров AVR. В памяти микроконтроллера загрузчик занимает всего лишь 207 слов, что является несомненным преимуществом. Однако он применяется в связке с памятью Atmel (AT45DB041B) и с фирменной энергонезависимой памятью Xilinx работать не будет. Кроме того, чтобы перенести его код на Cortex, потребуются некоторые усилия. Так что решение это – изящное и компактное, но имеет свои особенности и поэтому не универсально.

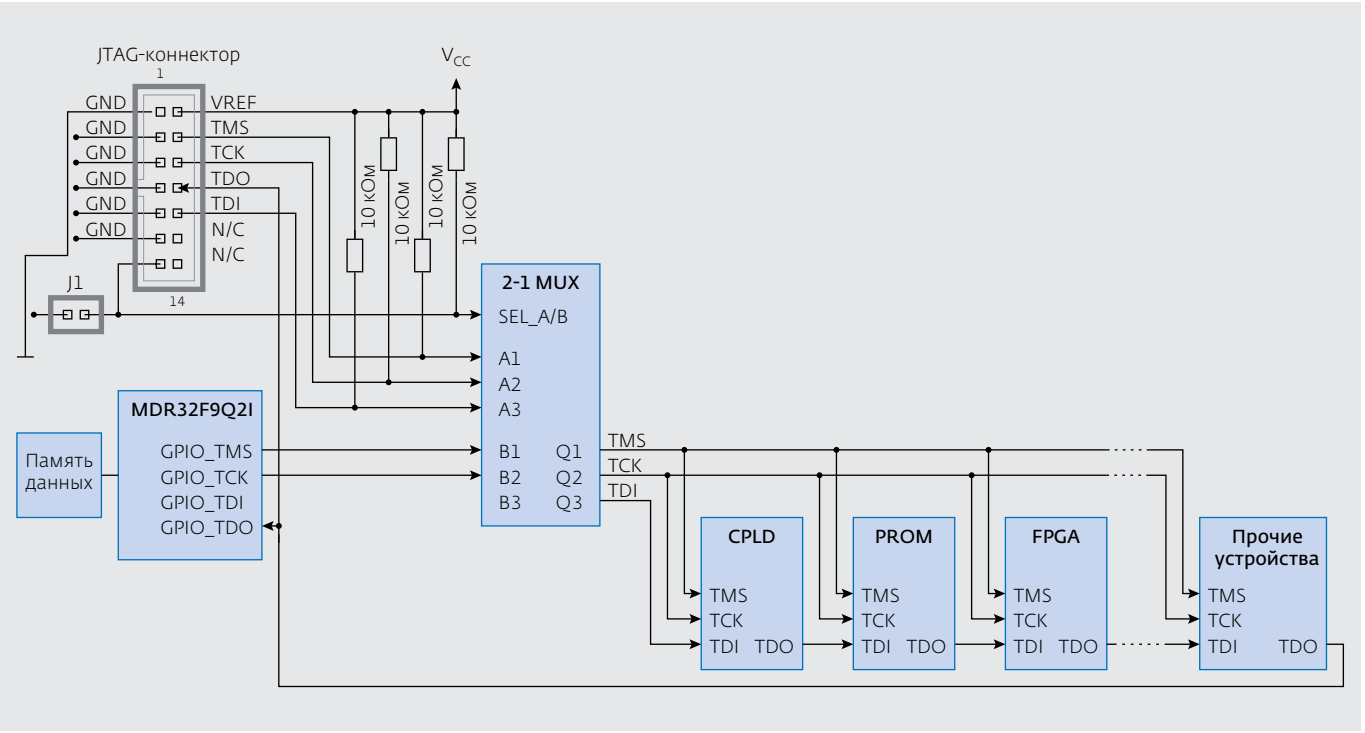

Рис.1. Аппаратная реализация внутрисхемного программирования ПЛИС и ПЗУ фирмы Xilinx с помощью встроенного микроконтроллера

Другой способ – конфигурирование ПЛИС Xilinx в режиме Slave Serial или SelectMAP [3] при помощи встроенного микроконтроллера. В этом случае, кроме микроконтроллера и программируемой ПЛИС (FPGA), в схеме появляется еще одна ПЛИС (CPLD), служащая интерфейсом между ними. Впрочем, если есть достаточно свободных выводов микроконтроллера, без CPLD можно обойтись, но следует помнить, что интерфейс программирования – синхронный параллельный, поэтому выводов понадобится много. Однако этот недостаток имеет и достоинство – высокую скорость конфигурирования ПЛИС (до 50 МГц и более). Стоит сказать и о том, что данный способ позволяет частично переконфигурировать ПЛИС, а также считывать ее текущую конфигурацию. Но наиболее интересный, на взгляд автора, способ конфигурирования ПЛИС с помощью встроенного микроконтроллера основан на использовании файлов формата *.xsvf [4].

Аппаратная реализация этого способа достаточно проста (рис.1) [4]. Конфигурация ПЛИС или ПЗУ загружается через интерфейс JTAG из микроконтроллера прямо на эти устройства, без какой-либо связующей логики. Передача данных выполняется с помощью стандартных сигналов JTAG-интерфейса: TDI (Test Data In – вход данных), TDO (Test Data Out – выход данных), TCK (Test Clock – тактовый сигнал порта тестирования) и TMS (Test Mode Select – выбор режима тестирования). Микроконтроллер может получать конфигурацию ПЛИС двумя путями: из подсоединенной к нему энергонезависимой памяти со стандартным интерфейсом для программирования либо от компьютера через любой интерфейс (например, USB или UART), контроллер которого есть в MDR32F9Q2I. Можно программировать ПЛИС и минуя контроллер – с помощью традиционного программатора Xilinx, подсоединяемого через JTAG-коннектор и связанного с компьютером через порты USB или LPT. Переключение между источниками конфигурационного файла выполняется мультиплексором. Если в цепочке много JTAG-устройств, то для обеспечения стабильной работы нужно очень внимательно следить за правильным распределением тактового сигнала TCK. Программирование ПЛИС по этой схеме проходит в несколько этапов: генерация конфигурационных файлов в формате XSVF, программирование микроконтроллера для интерпретации этих файлов, загрузка XSVF-файлов в микроконтроллер и их интерпретация и передача конфигурационных команд в ПЛИС через JTAG-интерфейс.

Формат XSVF основан на SVF (Serial Vector Format) – промышленном стандарте, который используется для описания работы с цепочкой JTAG-устройств [5]. В файлах SVF содержатся описания алгоритмов программирования устройств различных производителей в виде набора стандартных SVF-инструкций, а также информация, которая должна быть передана на сдвиговые регистры устройств в JTAG-цепочке для их программирования или тестирования. Каждое устройство, поддерживающее JTAG, имеет встроенный JTAG-контроллер, принимающий

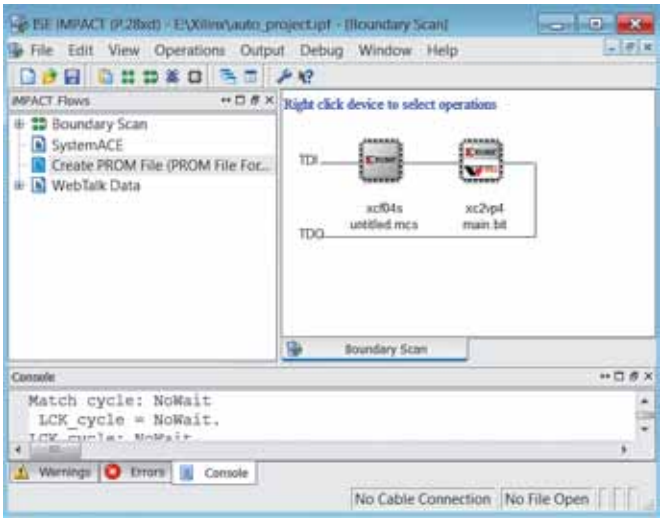

Рис.2. Окно программы Impact после сканирования JTAG-цепочки

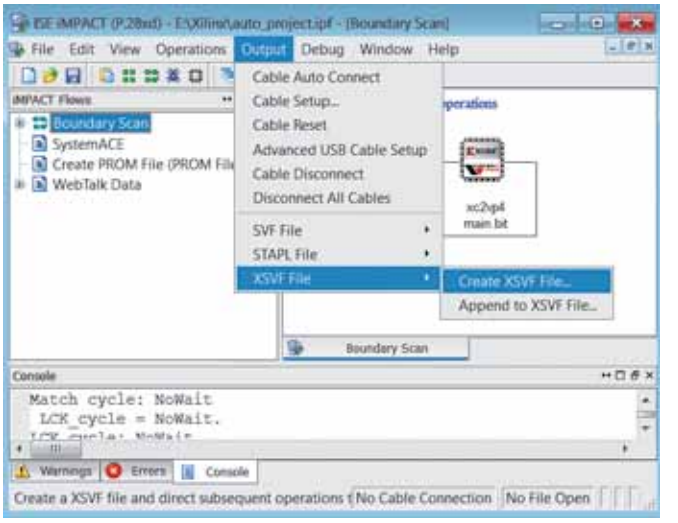

Рис.3. Настройка вывода информации в XSVF-файл в программе Impact

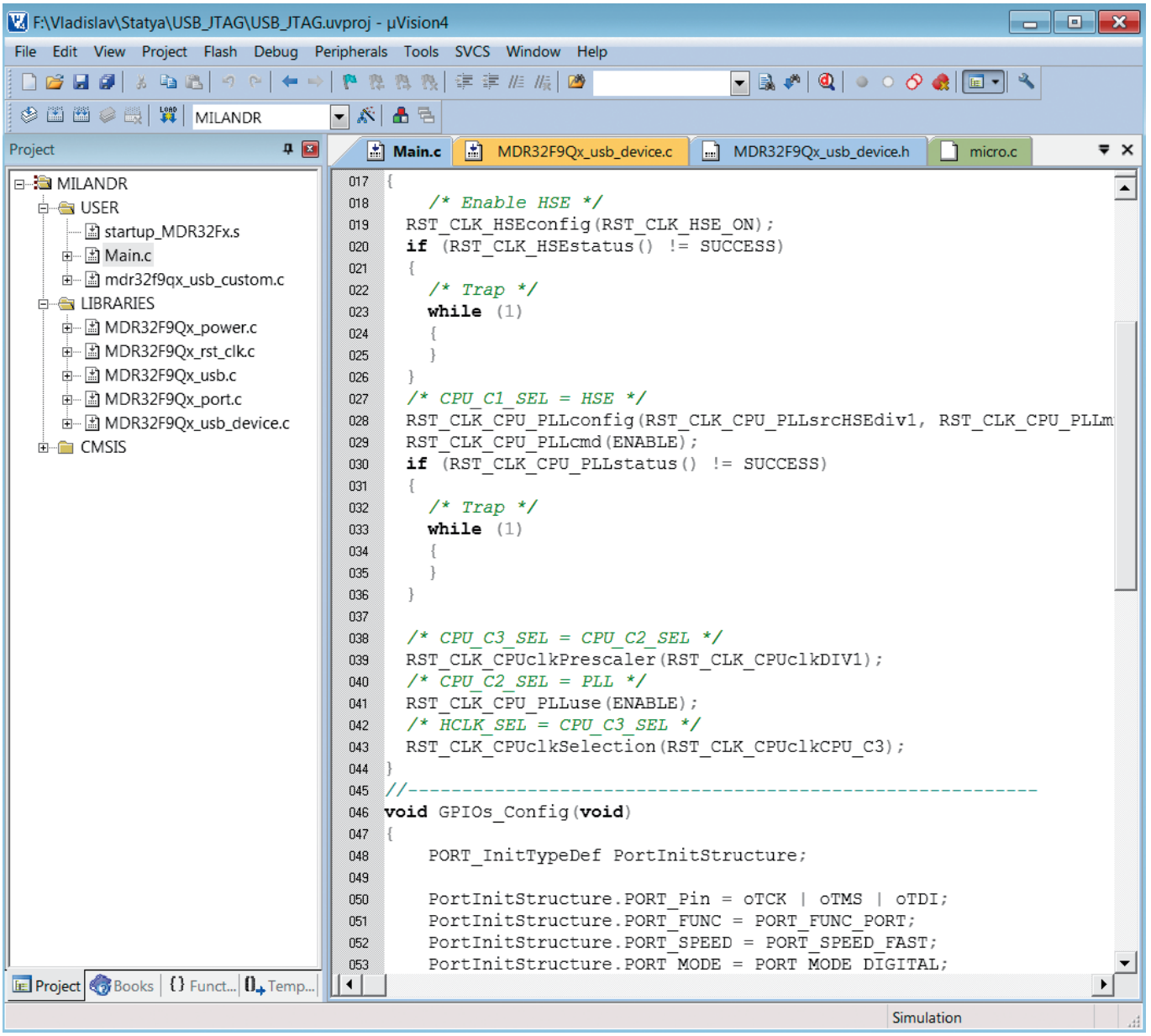

Рис.4. Проект-заготовка в среде Keil

и интерпретирующий эти команды. Файлы SVF имеют текстовый формат и занимают довольно много места. В файле XSVF (Xilinx SVF) команды представлены в бинарном виде и это существенно уменьшает объем занимаемой памяти.

Предположим, что у нас уже есть готовый проект для ПЛИС – конфигурационные файлы в форматах *.msc или *.bit. Конвертировать их в формат XSVF можно с помощью утилиты Impact, входящей в пакет Xilinx ISE. Перед запуском Impact нужно подключить ПЛИС к компьютеру через программатор Xilinx. Затем следует алгоритм действий, приведенный ниже:

- создать новый проект. Программа просканирует JTAG-цепочку через программатор и отобразит найденные устройства (рис.2);
- прикрепить к устройству из этой цепочки, которое мы хотим программировать, свой файл конфигурации (*.msc или *.bit), программа сама предложит это сделать. Для остальных устройств можно нажимать "Bypass";
- выбрать пункт меню "Create XSVF file" (рис.3) и задать имя файла для вывода;
- кликнуть правой кнопкой мыши на том устройстве из JTAG-цепочки, для конфигурирования которого мы хотим создать

XSVF-файл, и выбрать в появившемся меню пункт "Program";
- по окончании записи XSVF-файла (появится надпись "Program Succeeded") тем же путем, что и в третьем пункте, прекратить вывод данных в XSVF-файл, выбрав пункт "Stop writing to XSVF file".

Если заранее известно, какое устройство программируется, можно просто выбрать его в Impact. Программатор в этом случае не нужен.

Итак, XSVF-файл готов. Для того чтобы использовать его в микроконтроллере для программирования ПЛИС, необходима программа-интерпретатор, в качестве которой можно взять XSVF Player [4]. Ее исходный код состоит из нескольких файлов:
- Lenval.c, Lenval.h – содержат описание структуры для хранения XSVF-данных после считывания их с внешнего интерфейса контроллера и функции для работы с этой структурой;
- Ports.c, Ports.h – содержат функции для работы с портами микроконтроллера, которые будут служить выводами JTAG-интерфейса;
- Micro.c, Micro.h – содержат исходный код самого XSVF Player'a.

В исходном коде есть множество комментариев, вот только написан он для работы на компьютере через интерфейс LPT. Однако этот код можно перенести на любую желаемую платформу, если для нее есть среда разработки, поддерживающая язык Си. В случае MDR32F9Q2I есть даже возможность выбора среды, например, Keil (www.keil.com), IAR (www.iar.com) или CodeMaster-ARM (www.phyton.ru). Остановимся на первом, как считает автор, наиболее удобном и гибком решении. Стоит отметить, что среда Keil бесплатна при условии ограничения объема кода (32 Кбайт). Скачав ее с сайта производителя, создаем новый проект и выбираем в его свойствах тип контроллера (MDR32F2). На выходе имеем обычную заготовку проекта (рис.4), которую нужно доработать:
- добавить библиотеки для микроконтроллера MDR32F9Q2I (их можно найти на сайте производителя www.milandr.ru);
- отредактировать и добавить файлы исходных кодов из [4];
- создать и добавить файл main.c;
- инициализировать периферию (модуль тактирования, порты ввода-вывода);
- связать файлы исходных кодов из [4] и main.c в один рабочий проект и собрать его.

Прежде всего необходимо определиться с выводами микроконтроллера, которые будут задействованы для порта JTAG. Назначаем их в файле Ports.h (например, выводы 0–3 порта A можно определить как TCK, TMS, TDI, TDO, а сам порт A – как TAP). Далее нужно настроить модуль тактирования, а также проинициализировать те выводы микроконтроллера, которые будут использоваться как выводы порта JTAG (TCK, TMS, TDI – выходы, TDO – вход) для программирования ПЛИС. Эти процедуры стандартны, и функции для их выполнения находятся в фирменной библиотеке микроконтроллера MDR32F9Q2I.

Теперь можно подключать к проекту исходные коды из [4]. В файле Ports.c объявляется переменная out_word, содержащая значения логических уровней на выходных выводах порта JTAG (лог."0" или лог."1"). Там же находятся четыре функции, код которых зависит от платформы.

Первая функция – установка логических уровней на выводах порта JTAG. Параметр p выбирает один из выводов, параметр val – значение, которое должно быть на нем установлено (лог. "1" или лог. "0"):

```
void setPort(short p,short val)
{
  //смотрим, в какой вывод надо записать
  //новое значение
switch (p)
  {
    case TCK:
      out_word.bits.tck = val;
      break;
    case TMS:
      out_word.bits.tms = val;
      break;
    case TDI:
      out_word.bits.tdi = val;
      break;
    default:
      break;
  }
  //записываем новые данные
  //в порт интерфейса JTAG:
  //TAP – наш JTAG порт,
  //RXTX – аппаратный регистр
  //данных порта ввода-вывода
  //микроконтроллера
  TAP-> RXTX = out_word.value;
}
```

Вторая функция – генерация тактового импульса на выводе TCK. Здесь просто нужно два раза выставить логический уровень на

выводе TCK с помощью приведенной выше функции SetPort: сначала setPort(TCK,0), затем setPort(TCK,1).

Третья – чтение следующего байта XSVF-последовательности. Последовательность XSVF-команд может поступать на микроконтроллер посредством любого внешнего интерфейса (например, от компьютера по USB или UART).

Четвертая функция – чтение уровня (лог. "0" или лог. "1") с вывода TDO порта JTAG. Здесь считывается аппаратный регистр данных порта TAP и проверяется, выставлен ли нужный нам бит (TAP->RXTX & TDO).

Далее из файла Micro.c нужно убрать функцию int main(...), а также удалить директиву препроцессора #define DEBUG_MODE, чтобы выйти из отладочного режима работы программы. В файле Lenval.h устанавливается значение параметра MAX_LEN, регулирующего длину структуры для хранения XSVF-данных, равное 128. И, наконец, нужно добавить вызов функции xsvfExecute() (определена в файле Micro.c) в главный цикл программы микроконтроллера.

Теперь можно конфигурировать ПЛИС Xilinx, просто подключив ее к микроконтроллеру – остается только организовать его сопряжение с компьютером для загрузки XSVF-файлов. Проще всего это сделать через UART, наиболее целесообразно – использовать USB-контроллер, имеющийся в MDR32F9Q2I.

Опыт работы с микроконтроллером MDR32F9Q2I показал, что он очень удобен для разработчика. Подробная документация на русском языке не только к контроллеру, но и к его библиотекам – дополнительный стимул к разработке новых проектов на данной платформе.

ЛИТЕРАТУРА

1. Спецификация микроконтроллеров серий 1986BE9x, K1986BE9x и MDR32F9Qx. – ЗАО "ПКК "Миландр", 2012.
2. **O'Flynn C.** Robust Bootloader for FPGAs. – Circuit Cellar, 2006, №2.
3. **Peattie M.** Application Note: Virtex and Spartan FPGA Families. XAPP502. Using a Microprocessor to Configure Xilinx FPGAs via Slave or SelectMAP Mode. Xilinx, 2009.
4. **Kuramoto R.** Xilinx In-System Programming Using an Embedded Microcontroller. – Application Note: Xilinx Families. XAPP058, Xilinx, 2009.
5. **Bridgford B., Cammon J.** SVF and XSVF File Formats for Xilinx Devices. – Application Note: Xilinx Devices. XAPP503, Xilinx, 2009.