

Построение систем с процессором Microblaze на отладочной плате Nexys-4 в САПР Vivado

Евгений ГУРИН,
д. т. н., профессор
gurin2@yandex.ru

В статье рассматриваются вопросы создания систем с процессором Microblaze в САПР Vivado с применением нового инструмента IP Integrator. Процесс проектирования для ПЛИС Artix-7 показан на простейшем проекте, включающем ввод данных с переключателей и обмен по последовательному интерфейсу. Программное обеспечение реализуется в SDK. Приведены результаты проверки работоспособности созданной системы на отладочной плате Nexys-4.

Введение

В 2012 году фирма Xilinx выпустила САПР нового поколения Vivado [1–3], которая предназначена для замены в ближайшем будущем ISE Design Suite. Традиционно разработка систем с процессорными ядрами Microblaze в ISE Design Suite и в Vivado производится с использованием EDK. Начиная с версии 2013.3 в САПР Vivado добавлен IP Integrator [3–6], который также позволяет создавать системы с процессорными ядрами и описан в настоящей статье. В документации в качестве примера систем с процессорными ядрами Microblaze обычно рассматриваются системы на отладочных платах KC705 с ПЛИС Kintex 7 [3, 5, 6]. В то же время на практике неплохо себя зарекомендовали относительно недорогие и достаточно удобные в работе отладочные платы Nexys-4 фирмы Digilent [7]. Особенно удобно их применение в учебном процессе в вузах, где, как правило, проводятся несложные проекты и для подключения этих плат достаточно одного USB-кабеля. Настоящая статья может быть полезна преподавателям вузов, а также инженерам, занимающимся разработкой недорогих проектов на ПЛИС Artix 7. Статья ориентирована на специалистов, имеющих опыт создания простейших проектов в САПР Vivado.

Создание проекта и разработка аппаратной части

Сначала создается проект типа RTL в системе Vivado для ПЛИС XC7A100T 1CSG324C, используемый язык — VHDL. Последнее свойство можно задать при создании проекта либо вызвав Project Setting в окне Flow Navigator.

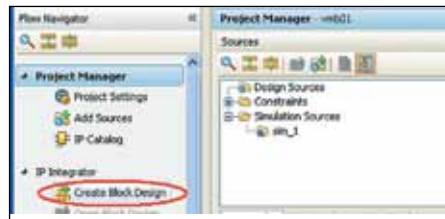


Рис. 1. Создание процессорного блока

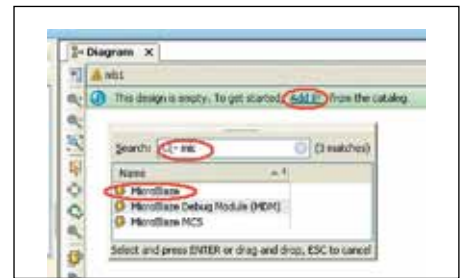


Рис. 2. Выбор процессора

Для создания процессорного блока в окне Flow Navigator активизируется строка **Create Block Design** (рис. 1), в открывшемся окне задается имя блока (например, mb1) и нажимается кнопка **OK**.

В открывшемся справа окне Diagram нажимается **Add IP**. В окне Search вводятся на-

чальные буквы нужного устройства (рис. 2) и двойным щелчком мыши выбирается строка **MicroBlaze**, в результате появляется изображение процессора. Если на этом изображении нажать двойным щелчком мыши,

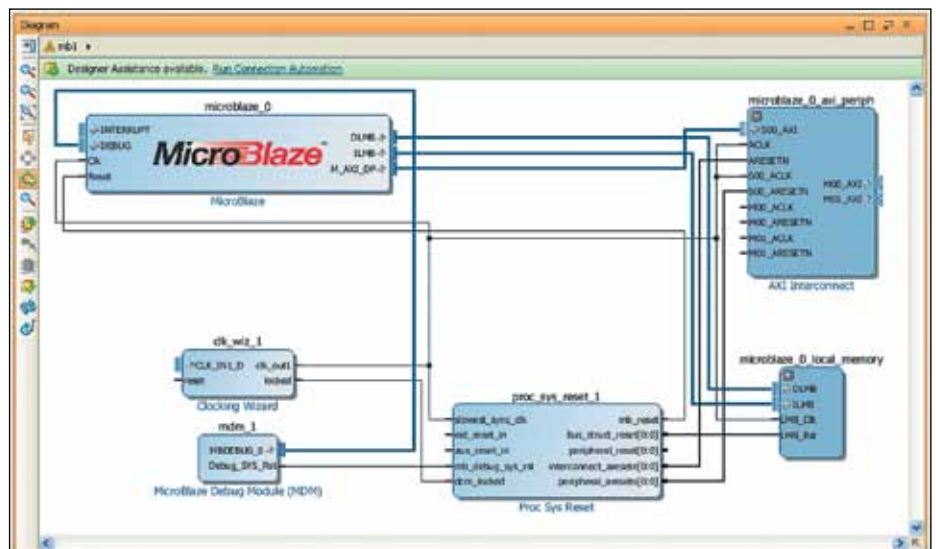


Рис. 3. Исходный вариант процессорного блока

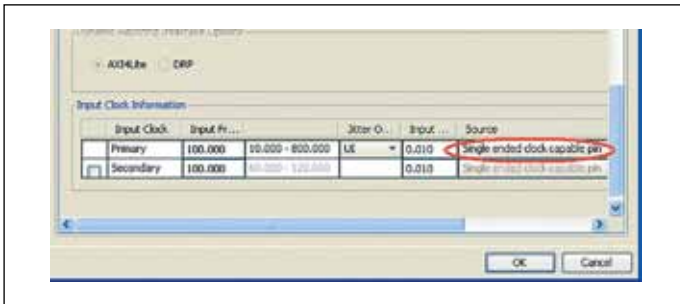


Рис. 4. Настройка устройства синхронизации



Рис. 6. Настройка устройства ввода-вывода

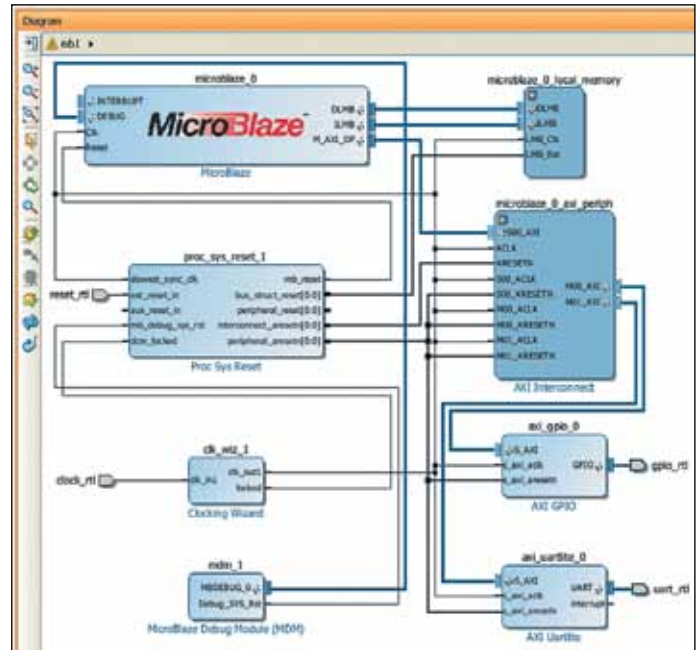


Рис. 8. Окончательный вариант процессорного блока

откроется первое из окон конфигурации процессора Re-customize IP (в исходной конфигурации четыре окна). При необходимости в конфигурацию процессора можно внести изменения, но в данном случае этого не нужно, а потому после ознакомления с содержимым окна следует нажать кнопку **Cancel**.

Для автоматического создания процессорного блока надо нажать **Run Block Automation** в окне **Diagram**, а затем выбрать единственный пункт `microblaze_0` в открывшемся меню. В результате появится окно, в котором следует увеличить локальную память Local Memory до 16 кбайт, остальные параметры остаются без изменений. Для продолжения нужно нажать клавишу **OK**, после чего в течение 15–20 с действует система проектирования, результат ее работы приведен на рис. 3.

Устройство `clk_wiz_1` по умолчанию генерируется с парафазным входом синхронизации `CLK_IN1_D`, в то же время на отладочной плате Nexys-4 вход синхронизации однофазный. Для изменения свойств `clk_wiz_1` надо нажать на нем двойным щелчком мыши, в результате откроется окно **Re-customize IP**. На закладке **Clocking Options**, которая активна после открытия окна, в поле **Input Clock Information** в верхней строке надо вместо **Differential clock capable pin** установить **Single ended clock capable pin**. Фрагмент этого окна с измененным параметром показан на рис. 4.

Не закрывая окна **Re-customize IP**, следует открыть закладку **Output Clocks**, в нижней части которой имеется поле **Enable Optional Inputs/Outputs**. В этом поле задаются дополнительные контакты `clk_wiz_1`. Здесь нужно убрать вход `reset`, после чего для закрытия окна нажимается **OK**. Стоит отметить, что вход сброса удалять необязательно,

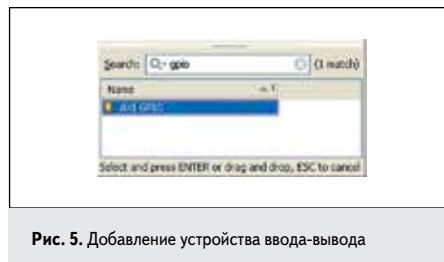


Рис. 5. Добавление устройства ввода-вывода

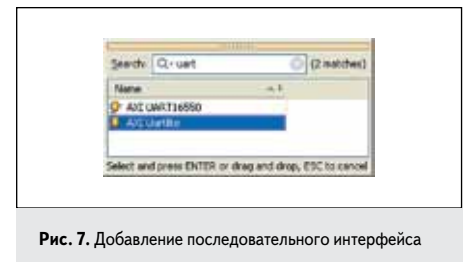


Рис. 7. Добавление последовательного интерфейса

альтернативный вариант со входом сброса рассмотрен в [3].

В созданный процессорный блок надо добавить периферийные устройства, пусть в данном проекте это будут восьмиразрядный вход с переключателями и последовательный интерфейс. Для добавления входа с переключателями на свободном месте окна **Diagram** необходимо щелкнуть правой клавишей мыши и в открывшемся меню выбрать пункт **Add IP**, затем в поле **Search** ввести `gpi0` (рис. 5) и нажать двойным щелчком мыши на строке `AXI_GPIO`.

На появившемся устройстве `AXI_GPIO` надо двойным щелчком мыши открыть окно свойств и изменить его содержимое, в частности, указать, что данное устройство является входным (**All Inputs**), и в поле **GPIO Width** задать разрядность 8 (рис. 6).

Аналогично добавляется последовательный интерфейс `AXI Uartlite` (рис. 7).

На появившемся устройстве надо двойным щелчком мыши открыть окно свойств и задать скорость (**Baud Rate**) 115 200.

Периферийные устройства добавляются в проект неподключенными. Для проведения связей надо щелкнуть мышью на строке **Run Connection Automation** в верхней части окна **Diagram**, в результате появляется перечень неподключенных контактов. Для проведения каждой связи надо ее выбрать и нажать **OK**

в открывшемся окне. При проведении сигнала сброса активный уровень (**ACTIVE_LOW**) изменять не нужно. При следующем нажатии на строке **Run Connection Automation** проведенные ранее связи из меню исключаются. После проведения последней связи строка **Run Connection Automation** исчезает.

Изображение блока и расположение составляющих его устройств формируются автоматически системой проектирования. При необходимости устройства можно передвинуть вручную, один из возможных вариантов изображения процессорного блока после такого передвижения приведен на рис. 8.

Созданный блок необходимо сохранить, например нажатием **Ctrl-s**. Затем создается главный файл проекта. Для этого в окне **Sources** правой клавишей мыши выделяется создан-

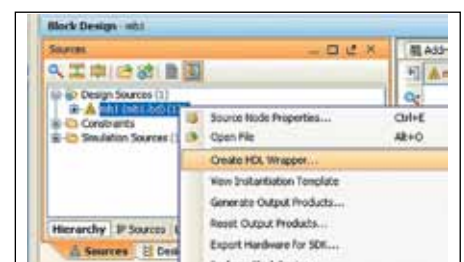


Рис. 9. Создание главного модуля проекта

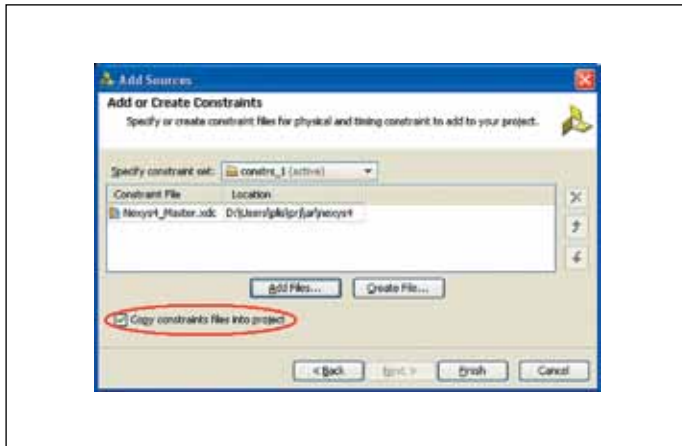


Рис. 10. Добавление файла ограничений

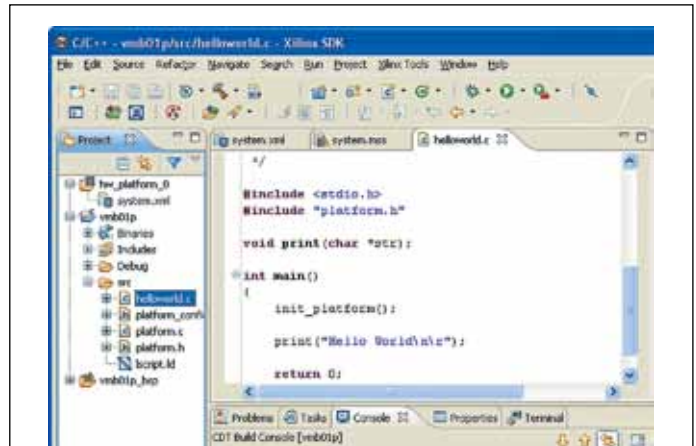


Рис. 12. Главное окно SDK

ный блок и в открывшемся контекстном меню выбирается пункт **Create HDL Wrapper** (рис. 9).

В следующем окне надо нажать **OK**. В результате создается главный модуль, имеющий следующее описание внешних сигналов:

```
entity mb1_wrapper is
port (
clock_rtl : in STD_LOGIC;
gpio_rtl_tri_i : in STD_LOGIC_VECTOR ( 7 downto 0 );
reset_rtl : in STD_LOGIC;
uart_rtl_rxd : in STD_LOGIC;
uart_rtl_txd : out STD_LOGIC
);
```

Далее необходимо задать ограничения. Для платы Nexys-4 на сайте производителя (www.digilentinc.com) доступен файл ограничений *Nexys4_Master.xdc*, который можно взять за основу. В окне **Flow Navigator** активизируется строка **Add Sources**, в появившемся окне выбирается **Add or Create Constraints** и нажимается кнопка **Next**. В следующем окне нажимается **Add Files**. В открывшемся окне указывается путь к файлу *Nexys4_Master.xdc* и нажимается **Enter** (или **OK**). В окне **Add Sources**, показанном на рис. 10, необходимо наличие флажка в поле **Copy constraints files into project** для того, чтобы в проекте была создана копия файла ограничений.

После нажатия на кнопку **Finish** файл ограничений создается в проекте, его необходимо открыть и изменить в нем имена в соответствии с именами в заголовке модуля *mb1_wrapper*, приведенном выше. Сначала необходимо убрать комментарии в строках 8–10 файла *Nexys4_Master.xdc* и заменить в них *clk* на *clock_rtl*, запись *sys_clk_pin* в строке 10 остается неизменной. В окончательном варианте эти строки будут иметь следующий вид:

```
set_property PACKAGE_PIN E3 [get_ports clock_rtl]

set_property IOSTANDARD LVCMOS33 [get_ports clock_rtl]

create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports clock_rtl]
```

Следует учесть, что строки в файле ограничений достаточно длинные и не всегда

умещаются на одной строке в настоящем тексте. Так, например, в приведенном выше фрагменте строка 10 файла ограничений расположена в двух последних строках. Кроме того, большинство строк начинается с ключевой фразы *set_property*. Исключение в данном примере составляет только строка 10, которая начинается фразой *create_clock*.

Затем в файле ограничений *Nexys4_Master.xdc* необходимо убрать одиночные комментарии в строках 13–36 (двойные комментарии остаются). В этих строках имена *sw* заменяются на *gpio_rtl_tri_i*. Для переключателя *sw0* строки 14 и 15 в файле ограничений будут иметь следующий вид:

```
set_property PACKAGE_PIN U9 [get_ports {gpio_rtl_tri_i[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {gpio_rtl_tri_i[0]}]
```

Аналогично будут выглядеть записи для *sw7...sw1*, отличаться будут только индексы.

Пусть сигнал сброса поступает с кнопки **CPU RESET**. Для этого в строках 191–192 файла *Nexys4_Master.xdc* необходимо убрать комментарии и заменить *btnCpuReset* на *reset_rtl*. Окончательный вид этих строк будет следующим:

```
set_property PACKAGE_PIN C12 [get_ports reset_rtl]

set_property IOSTANDARD LVCMOS33 [get_ports reset_rtl]
```

Для последовательного интерфейса надо убрать комментарии в строках 487–488 и 490–491 в файле *Nexys4_Master.xdc* и заменить имена *RsRx* и *RsTx* на *uart_rtl_rxd* и *uart_rtl_txd* соответственно. В окончательном варианте строки будут иметь следующий вид:

```
set_property PACKAGE_PIN C4 [get_ports uart_rtl_rxd]

set_property IOSTANDARD LVCMOS33 [get_ports uart_rtl_rxd]

## ...

set_property PACKAGE_PIN D4 [get_ports uart_rtl_txd]

set_property IOSTANDARD LVCMOS33 [get_ports uart_rtl_txd]
```



Рис. 11. Запуск SDK

После изменений файл ограничений надо сохранить.

Далее выполняется синтез проекта, для чего в окне **Flow Navigator** активизируется строка **Run Synthesis**. Затем выполняется реализация проекта (**Run Implementation**) и генерация битового (загрузочного) файла (**Generate Bitstream**).

Разработка программного обеспечения

После создания аппаратной части можно приступить к созданию программы. Для этого из основного меню выполняется команда **File** → **Export** → **Export Hardware for SDK**. В открывшемся окне необходимо установить флажок **Launch SDK** (рис. 11) и нажать **OK**.

В результате открывается основное окно **SDK**. Для создания проекта из основного меню выполняется команда **File** → **New** → **Application Project**. В открывшемся окне задается имя проекта (например, *vmb01p*), остальные параметры не изменяются. Для перехода к следующему окну нажимается клавиша **Next**. В окне **Templates** выбирается **Hello World** и нажимается клавиша **Finish**. Фрагмент основного окна созданного проекта показан на рис. 12. В нем в качестве стандартного устройства ввода-вывода автоматически назначается последовательный интерфейс *axi_uartlite_0*.

После этого проект можно записывать в плату. Для этого из основного меню SDK выполняется команда **Xilinx Tools** → **Program FPGA**. В результате открывается окно, показанное на рис. 13. В этом окне надо указать путь к загрузочному файлу с расширением *.bit* и файлу с расширением *.bmm*, а также указать, что загружается программа *vmb01p.elf*. Перед загрузкой плату необходимо подключить к компьютеру кабелем *micro-USB* и включить питание.

Перед запуском программы на решение необходимо активизировать терминал, а для этого в окне **Terminal** — нажать кнопку **Connect** (рис. 14). Терминал должен быть настроен на работу со скоростью 115 200, возможный вариант настройки показан на рис. 15.

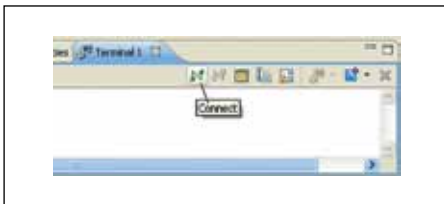


Рис. 14. Включение терминала

Для запуска программы на решение в окне **Project** надо правой клавишей мыши выделить проект *vmb01p* и из открывшегося меню выполнить команду **Run As** → **Launch on Hardware**. В результате работы программы командой `print` в стандартное устройство вывода будет отправлено сообщение **Hello World**, которое передается в компьютер и будет отображено в окне терминала.

На рис. 16 приведен пример работы программы **Hello World** при использовании терминала SDK. Отображение результатов работы возможно также программой **Tera Term** [8].

Изменим программу, пусть она будет иметь следующий вид:

```
#include "platform.h"
#include <stdio.h>
#include "xgpio_1.h"
#include "xparameters.h"
int main()
{ int i, dd=0, sw=0;
  init_platform();
  xil_printf("\n\r vmb01 begin \n\r");
  while(1) {
    sw=Xil_In32(XPAR_GPIO_0_BASEADDR); //read data from
  swithes
    if(sw != dd)
      xil_printf("\n\r sw=%d", sw); //new data
    for(i=0; i<500000; i++) // delay
      dd=sw; // delay data
  }
  return 0;
}
```

С помощью функции **Xil_In32** данные считываются с переключателей на отладочной плате и присваиваются переменной *sw*. Константа *XPAR_GPIO_0_BASEADDR* определена в файле *xparameters.h*. Определение функции **Xil_In32** находится в файле *xil_io.h*, ссылка на который имеется в файле *xgpio_1.h*.

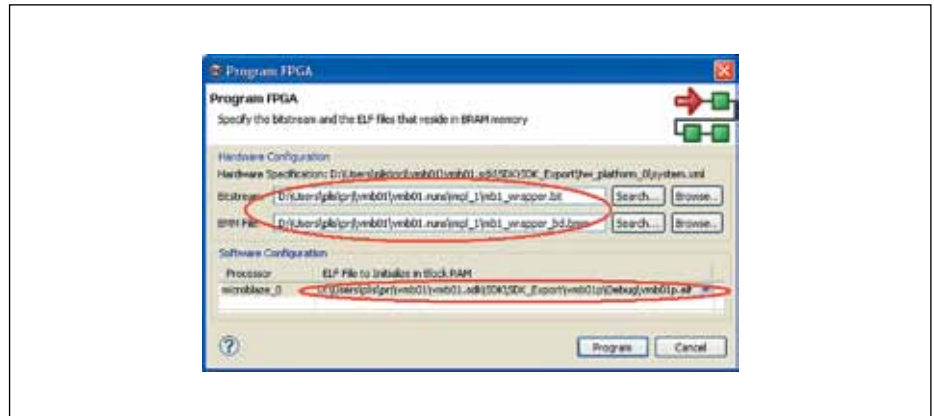


Рис. 13. Программирование ПЛИС



Рис. 16. Результат работы программы Hello World

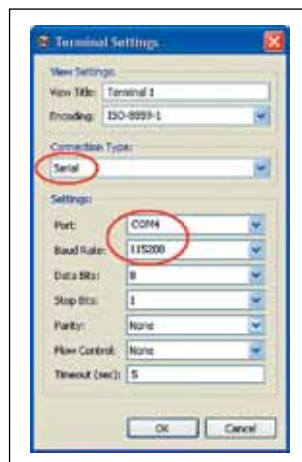


Рис. 15. Окно настройки терминала

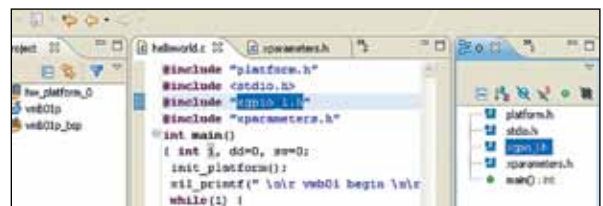


Рис. 17. Новый вариант программы

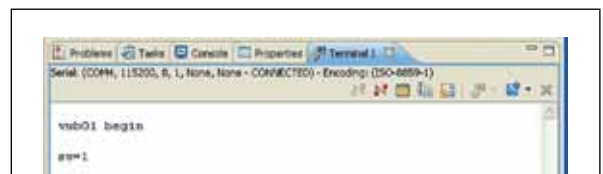


Рис. 18. Результат работы системы

Открыть файл *xgpio_1.h* можно двойным щелчком мыши на его имени в правой части основного окна SDK (рис. 17). После открытия файла *xgpio_1.h* из него таким же образом открывается файл *xil_io.h*. Следует отметить, что для ввода данных могут использоваться и другие функции.

В следующей строке программы происходит сравнение нового значения *sw* и старого (задержанного) значения *dd*. Если они не равны (то есть состояние переключателей изменилось), то *sw* передается по последовательному интерфейсу функцией **xil_printf**. В следующих двух строках с помощью оператора **for** реализована задержка для исключения дребезга на переключателях.

Перед запуском программы переключатели *sw7...sw0* на отладочной плате Nexys-4

должны быть установлены в нижнее положение. Если активен терминал SDK, то после запуска программы в его окне будет выдана запись *vmb01 begin*.

Если переключатель *sw0* на отладочной плате Nexys-4 установить в верхнее положение, в окне терминала SDK будет выдана запись *sw=1* (рис. 18). Далее после каждого изменения положения переключателей *sw7...sw0* в окне терминала будет выводиться их новое состояние в десятичной системе счисления.

В статье рассматривалась отладочная плата Nexys-4. Однако представленная выше последовательность действий может использоваться и во многих других похожих случаях при создании систем на основе процессора Microblaze.

Литература

1. Тарасов И. Е. Маршрут проектирования ПЛИС Xilinx в САПР Vivado // Компоненты и технологии. 2012. № 12.
2. Зотов В. Ю. Конвертирование проектов цифровых устройств, разрабатываемых на основе ПЛИС и полностью программируемых систем на кристалле фирмы Xilinx в среде ISE Design Suite, в формат САПР Vivado Design Suite // Компоненты и технологии. 2013. № 8–10.
3. Vivado Design Suite. User Guide. Embedded Processor Hardware Design // Xilinx, UG898 (v2013.3), October 2, 2013.
4. Тарасов И. Е. Использование IP Integrator в САПР Vivado для ПЛИС серии 7 и UltraScale // Компоненты и технологии. 2013. № 12.
5. Vivado Design Suite. User Guide. Designing IP Subsystems Using IP Integrator. Xilinx, UG994, October 2, 2013.
6. Vivado Design Suite Tutorial: Embedded Processor Hardware Design. UG940, October 2, 2013.
7. www.digilentinc.com
8. Тарасов И. Е. Проектирование в САПР EDK на базе All Programmable SoC семейства Zynq 7000 // Компоненты и технологии. 2012. № 12.