Жармакин Б.К., магистр
Корганбаева Л.Н., магистр
Садиев Нуржан., студент
ЕНУ им. Л.Н.Гумилева

Программирование постоянной яркости светодиода на Spartan-3E Starter Kit

Программируемая логическая интегральная схема (ПЛИС, англ. programmable logic device, PLD) — электронный компонент, используемый для создания цифровых интегральных схем. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования (проектирования). Для программирования используются программаторы и отладочные среды, позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры: Verilog, VHDL, AHDL и др. Альтернативой ПЛИС являются: программируемые логические контроллеры (ПЛК), базовые матричные кристаллы (БМК), требующие заводского производственного процесса для программирования; ASIC — специализированные заказные большие интегральные схемы (БИС), которые при мелкосерийном и единичном производстве существенно дороже; специализированные компьютеры, процессоры (например, цифровой сигнальный процессор) или микроконтроллеры, которые из-за программного способа реализации алгоритмов в работе медленнее ПЛИС.

Некоторые производители ПЛИС предлагают программные процессоры для своих ПЛИС, которые могут быть модифицированы под конкретную задачу, а затем встроены в ПЛИС. Тем самым обеспечивается уменьшение места на печатной плате и упрощение проектирования самой ПЛИС, за счёт быстродействия

Семейство **Spartan-3E** специально разработано для использования в электронных устройствах, рассчитанных на большие тиражи и недорогие комплектующие. Всего в семейство входит пять кристаллов, различающихся логической ёмкостью,

при этом минимальный по ёмкости кристалл содержит 100 тыс. эквивалентных системных вентилей, а максимальный - 1,6 млн.

Новый инструментальный комплект **Spartan-3E Starter Kit** предназначен в первую очередь для практического изучения проектирования цифровых устройств с аппаратной реализацией операций и разработки встраиваемых микропроцессорных систем **FPGA** фирмы **Xilinx**. Уникальные функциональные возможности, технические характеристики и конструктивное исполнение инструментального модуля, входящего в этот компонент, позволяют не только выполнять отладку проектов, реализуемых на основе ПЛИС семейства **Spartan-3E**, но и и использовать его в качестве промышленного серийного варианта разрабатываемого устройства.

Начнем знакомство с возможностями **Spartan-3E Starter Kit** с самого, пажалуй, простого. А именно с программирования постоянной яркости светодиода на даташите ПЛИС.

Яркость светодиода можно регулировать подачей на него разных значений постоянного напряжения (например, переменным резистором). Но на нашей плате установлены светодиоды без переменных резисторов, которые могут принимать значение **«1»** и светиться в полную яркость, либо **«0»**. Так как же тогда сделать регулировку яркости у такого столь простого девайса? Ответ – ШИМ. Вся суть в том, что плата будем «моргать» этим светодиодом настолько быстро, что моргание даже не будет заметно нашему глазу, а мы просто будем видеть тускло светящийся светодиод. Если сказать точнее, то просто у светодиода есть переходный процесс при зажигании, то есть загорается он не мгновенно. Именно этим мы пользуемся, подавая единицу на очень короткий промежуток времени, так чтобы светодиод не успел загореться в полную яркость.

Загружаем **ISE Project Navigator** и нажимаем **File -> New Project.** Пишем имя проекта, выбираем директорию для сохранения и снизу выбираем **Top-level source type: *HDL***. Далее *Next*.

Далее выбираем ПЛИС. В нашем случае ***Family: Spartan3E; Device: XC3S500E; Package: FG320; Speed: - 4.*** Все эти данные можно увидеть на самой микросхеме, либо посмотреть в даташите (смотрите рисунок ниже).

Далее выбираем *Preferred Language: VHDL*, жмем *Next* и потом *Finish*. Проект создан. Теперь нужно добавить в него модуль, в котором мы будем описывать логику работы светодиода. Слева вверху находим кнопку *New Source* и нажимаем на нее.

В появившемся окне выбираем **VHDL Module** и пишем любое имя файла (можно одноименное с проектом). Нажимаем *Next*.

Теперь нам нужно задать используемые в проекте ножки. Этого можно не делать сейчас и пропустить этот шаг, а потом написать все руками. Но поскольку для нашего проекта нам потребуется всего лишь три ножки, то я ввел их прямо здесь. Итак, нам потребуется опорная частота от установленного на плате 50 МГц кварца, подключенного к ПЛИС к ножке с именем **C9**, а так же мы будем использовать два светодиода так же уже установленных на плате. Допустим, это будут два правых светодиода, подключенных к ножкам ПЛИС под именами **E12** и **F12.** Назовем ножку кварца **clk**, установим **Direction: IN** т.к. мы будем считывать частоту, а ножки со светодиодами – **led1** и **led2** со значением **Direction: OUT**, т.к. мы будем ими управлять.
Нажимаем *Next*, потом *Finish*. Видим открывшийся текстовый редактор с уже заполненной заготовкой I/O портов проекта.

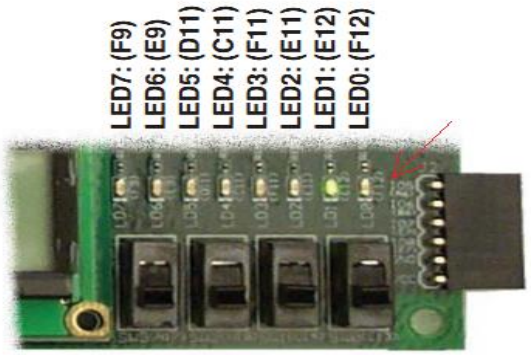
```
30  entity sharb is
31      Port ( clk : in  STD_LOGIC;
32             led1 : out  STD_LOGIC;
33             led2 : out  STD_LOGIC);
34  end sharb;
35
36  architecture Behavioral of sharb is
37
38  begin
39
40
41  end Behavioral;
```

Можно было пропустить предыдущий шаг и ввести все это вручную. Далее нам нужно сопоставить имена портов с именами ножек ПЛИС. Нажимаем правой кнопкой на имени файла **sharb.vhd** в иерархии и выбираем пункт *New Source*.

В открывшемся окне выбираем **Implementation Constrains File** и называем этот файл **pin.** Нажимаем *Next*, затем *Finish*.
В открывшийся пустой файл пишем следующее:

```
NET "clk" LOC = "C9";
NET "led1" LOC = "F12";
NET "led2" LOC = "E12";
```

Номера ножек можно посмотреть в даташите, либо в нашем случае нам упростили поиск — номера можно посмотреть прямо на плате рядом с нужной периферией:



Переключаемся в файл **sharb.vhd** и пишем код:

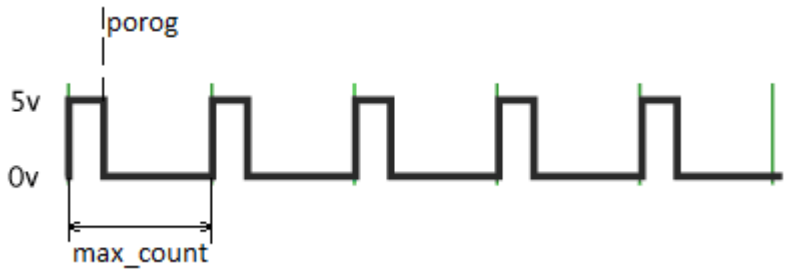
```
architecture Behavioral of shabr is
    constant clk_freq  : integer := 50_000_000; -- частота кварца
     constant shim_freq : integer := 10_000;     -- частота ШИМ
      constant max_count : integer := clk_freq / shim_freq; -- разрядность ШИМ
signal count: integer range 0 to max_count := 0; -- счетчик делителя частоты
 constant porog: integer := max_count / 48; -- ширина импульса логической единицы
begin
  process(clk)
     begin
      if rising_edge(clk) then
        if count = max_count then
          count <= 0;
        else
```

```
        count <= count + 1;
      end if;
    end if;
  end process;
      led1 <= '1' when count < porog else '0';
            led2 <= '1';
      end Behavioral;
```

Теперь разберемся, что делает этот код. Вначале обратите внимание на строчку **led2 <= '1'.** Мы зажигаем второй светодиод в полную яркость, подавая туда логическую единицу, чтобы нам было с чем сравнивать яркость свечения первого светодиода. Далее смотрим на объявленные регистры и константы. Константа **clk_freq** хранит частоту кварца в герцах; **shim_freq** это частота ШИМ в герцах. Соответственно, чтобы получить нужный нам период ШИМ, необходимо поделить тактовую частоту на частоту ШИМ, и мы получим число тактов главного кварца соответствующее периоду ШИМ. По сути это будет являться разрядностью ШИМ. Результат деления записываем в константу **max_count.** Далее создаем счетчик **count,** который будет циклично считать от **0** до **max_count** на частоте 50МГц. Создаем процесс **process(clk).** Условие **if rising_edge(clk) then** ждет очередного «тика» с кварца, и если он произошел выполняет прибавление на единичку счетчика **count,** проверяя не досчитал ли он до максимального значения. Далее, вне процесса пишем строку

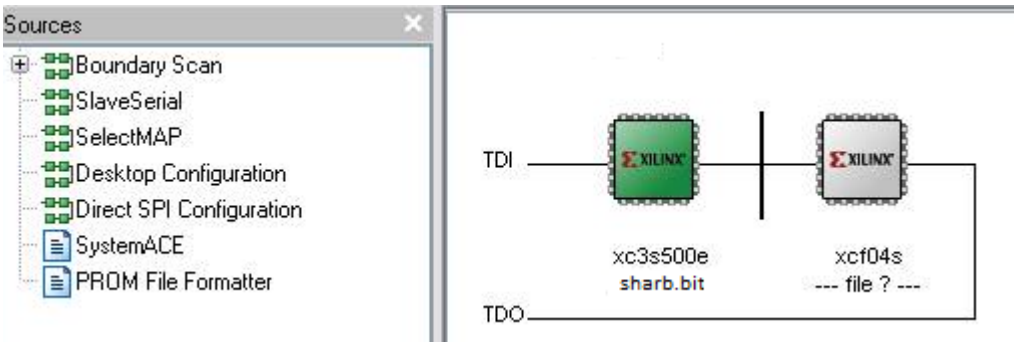
```
led1 <= '1' when count < porog else '0';
```

То есть на светодиоде висит логическая единица тогда, когда наш счетчик меньше какого-то порогового значения **porog** (в нашем случае это 1/48 от всего периода, смотри объявление констант), остальную часть периода на светодиоде висит логический ноль. Это наглядно можно показать рисунком:



Сохраняем все изменения, выбираем в иерархии файл **sharb.vhd** и снизу под иерархией ищем процесс *Configure Target Device* и запускаем его. Ждем, пока проект оттранслируется в файл прошивки, после чего откроется окно программы **iMPACT**, с помощью которой мы будем зашивать его в ПЛИС.

Двойной щелчек на *Boundary Scan*, и если у вас плата подключена к компьютеру по USB, то вы увидите примерно следующее:



Если вам не предложили выбрать файл прошивки для **xc3s500e,** то кликнем правой кнопкой по соответствующей микросхеме и выбирайте пункт меню *Assign Configuration File*. В окне выбора файла выбираем недавно созданный **sharb.bit**. Далее опять правой кнопкой на **xc3s500e,** затем *Program*. Запустится процесс зашивки, после чего появится надпись *Program Successful*. Если все прошло именно так, то можно смотреть на плату и наслаждаться первыми результатами на этом сложном пути программирования интегральных схем.

Используемая литература:

1. Зотов В.Ю. Проектирование цифровых устройствна основе ПЛИС фирмы XILINX в САПР WebPACK ISE. – М.: Горячая линия – Телеком. 2003. – 624 с.
2. Тарасов И.Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL
3. http://fpga-faq.narod.ru
4. http://electronix.ru/forum/
5. http://kit-e.ru/articles/cad/2006_10_64.php
6. http://ru.wikipedia.org/wiki/JTAG
7. http://habrahabr.ru/post/133383/

Публикация:

ӘОЖ 621.3
КБЖ 32 978-601-301-193-6

Е40 II Халықаралық мектеп – семинар «ӨЖЖ және ҚЖЖ техникасы мен технологиялары» Бурабай, 23-25 маусым 2014 жыл – Астана: Л.Н. Гумилев атындағы ЕҰУ, 2014. – 258 б.

Е40 II Международная летняя школа- семинар «Техника и технологии СВЧ и КВЧ» 23-25 июня 2014 года, Боровое – Астана: ЕНУим Л.Н. Гумилева, 2014. – 258с.

THE 2-ND INTERNATIONAL SUMMER SCHOOL SEMINAR «Technique and Technologies of Extremely and Super-High Frequencies» 23.06.2014-25.06.2014