ПЛИС IGLOO2 корпорации Microsemi: практикум по сверхнизкому энергопотреблению

Дмитрий ИОФФЕ
support@actel.ru
Артём КАЗАКОВ
kazakov@actel.ru

Одна из наиболее сильных сторон новых ПЛИС корпорации Microsemi IGLOO2 и SmartFusion2 — развитая система снижения энергопотребления до чрезвычайно малых величин, вплоть до 1 мВт. В предлагаемой статье, основанной на специальном руководстве корпорации Microsemi [1], описано практическое исследование работы этой системы на базе недорогого отладочного набора IGLOO2 Evaluation Kit.
Данный материал служит продолжением цикла, посвященного ПЛИС IGLOO2 и SmartFusion2 [2, 3] и опубликованного в журнале «Компоненты и технологии».

Введение

Современные программируемые логические интегральные схемы (ПЛИС) очень сложные устройства. Они содержат не только собственно массив программируемой логики (programmable gate array), но и множество аппаратных узлов, расширяющих их возможности. Это скоростные преобразователи параллельного кода в последовательный и обратно (SERDES), подсистемы управления памятью и многое другое, вплоть до микроконтроллерных ядер. Новейшие ПЛИС корпорации Microsemi, представленные семействами IGLOO2 и SmartFusion2, пожалуй, даже выделяются на общем фоне обилием и разнообразием встроенных аппаратных блоков, которые наделяют их большим количеством функций и позволяют сократить число внешних компонентов.

Сегодня мы познакомимся с одной из самых интересных подсистем ПЛИС семейства IGLOO2 — подсистемой энергосбережения. Благодаря этой подсистеме новые ПЛИС корпорации Microsemi отличаются своей экономичностью среди весьма продвинутых в этом отношении конкурентов. Знакомство состоится по ходу лабораторной работы, которую мы выполним по фирменному руководству [1] с использованием отладочного набора IGLOO2 Evaluation Kit [3].

Режим сверхмалого энергопотребления Flash*Freeze

Flash*Freeze — это режим ожидания с ультранизким статическим потреблением вплоть до 1 мВт, в который устройство может легко входить и выходить. Таким образом, когда от прибора не требуется активная деятельность, потребляемая мощность сокращается в десятки раз. Такой режим находит применение в мобильных девайсах, в медицинских системах для наблюдения за пациентами и во многих других случаях, где необходима минимальная статическая потребляемая мощность.

Устройство можно перевести в режим Flash*Freeze по сигналу извне или по внутреннему сигналу. Выход из режима происходит по наличию любой активности на линиях ввода/вывода или же при появлении на них заданной сигнатуры (конфигурируется пользователем). Причем внешний источник тактирующего сигнала не нужен. Подсистема памяти HPMS (High Performance Memory Subsystem) остается включенной, поэтому данные в статическом ОЗУ сохраняются. Также сохраняется информация в регистрах и сведения о состоянии линий ввода/вывода, соответственно, возврат в активный режим происходит очень быстро.

Более подробно режим Flash*Freeze описан в тексте [4].

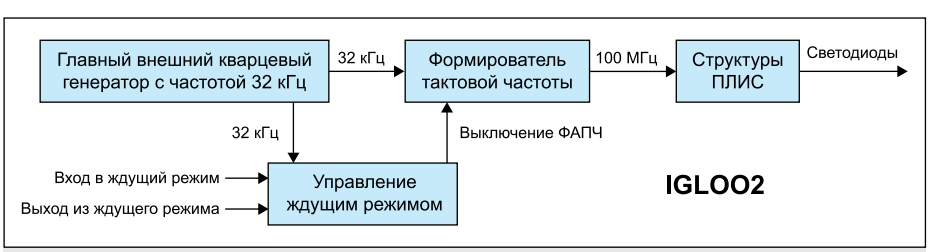

Рис. 1. Структурная схема отладочного проекта

Подготовка к работе

По ходу работы мы создадим и откомпилируем специальный отладочный проект (рис. 1), который содержит большое количество непрерывно переключающихся триггеров: 421 18-разрядный двоичный счетчик, 604-разрядный сдвиговый регистр, 11 блоков LSRAM и математические блоки. Формирователь тактовой частоты (Fabric clock conditioning circuit, FCCC) при помощи системы ФАПЧ (англ. PLL — Phased-locked loop) генерирует частоту 100 МГц из входной частоты 32 кГц, поступающей от внешнего кварцевого генератора. Сигнал захвата (Lock) ФАПЧ используется для сброса ПЛИС. Схема управления переходом в ждущий режим содержит тактируемый RS-триггер, который включает и выключает питание ФАПЧ в формирователе тактовой частоты. Кроме того, проект содержит блок управления светодиодами, отображающими состояние ПЛИС при входе в ждущий режим и выходе из него.

Файлы для создания проекта необходимо скачать с сайта корпорации Microsemi [5].

Для этого надо войти в раздел ASIC, FPGA & SoC и воспользоваться в меню следующим маршрутом: ASIC, FPGA & SoC → Design Resources → Dev Kits → IGLOO2 → Evaluation Kit. Откроется страница, озаглавленная IGLOO2 Evaluation Kit, на которой надо открыть вкладку **Training & Demos**. Здесь мы найдем большую подборку обучающих материалов по IGLOO2, в частности несколько лабораторных работ для изучения различных свойств семейства. Нас сейчас интересует группа материалов под заголовком Low Power. В ней есть ссылка на руководство IGLOO2 FPGA Low Standby Power — Libero SoC v11.4 Demo Guide [1], которое нам понадобится, а чуть ниже расположена ссылка на архив с файлами проекта к нему — Design Files. Скачаем и то и другое. В тексте руководства имеются ссылки на другие материалы, весьма полезные для углубленного изучения.

Кроме того, с сайта Microsemi надо скачать САПР Libero SoC версии 11.4 и установить ее, если это еще не сделано. О последовательности действий подробно рассказано в статье [6], там же есть описание главного окна среды Libero SoC и порядок работы с ней. Заметим, что в разделе системных требований к лабораторной работе указана 64-разрядная ОС семейства Windows. Тем не менее проект, описанный в настоящей статье, создан на компьютере с 32-разрядной ОС Windows XP SP3 и 2 Гбайт ОЗУ. То есть на начальном этапе ознакомления с семейством IGLOO2 можно использовать недорогой и не самый новый компьютер.

В [1] подразумевается, что лабораторная работа будет выполняться на отладочной плате из набора IGLOO2 Evaluation Kit. Кстати, проект из данной работы можно использовать в качестве готового блока управления режимом сверхнизкого потребления в целевом устройстве. Набор IGLOO2 Evaluation Kit стоит недорого, а после завершения ознакомления с ПЛИС IGLOO2 его можно употребить с пользой — например, превратить в программируемый лабораторный генератор сигналов. А лет через десять достать его из стола и вздохнуть: как далеко ушла вперед техника…

Итак, приступаем. Распакуем скачанный архив с файлами *IGL2_Standby_tutorial_11p4_DF.zip* в заготовленную под наши учебные проекты папку на жестком диске и начнем.

Отметим, что по ходу работы нам может понадобиться подключение к Интернету.

Создание проекта

Запустим среду проектирования Libero SoC. Из меню **Project** выберем пункт **New Project**. Появится диалоговое окно **New Project** (рис. 2).

Заполним поля этого окна так, как показано на рис. 2:

- Project Name: IGL2_Standby;
- Project Location: папка IGL2_Standby внутри распакованного содержимого архива, где лежит файл нашего проекта IGL2_Standby.prjx. Например, D:\Microsemi_prj\IGL2_Standby_tutorial\Libero_project\IGL2_Standby;
- Preferred HDL type: VHDL;
- Family: IGLOO2;
- Die: M2GL010T;
- Package: 484 FBGA;
- Speed: -1;
- Core Voltage (V): 1.2;
- Ramp rate: 100ms Minimum;
- Junction Temperature: COM;
- Core Voltage: COM;
- PLL Supply Voltage (V): 3.3 (обратите внимание: по умолчанию стояло 2,5 В, но в отладочной плате подано 3,3 В);
- Use Design Tool: снять флажок (по умолчанию он был установлен).

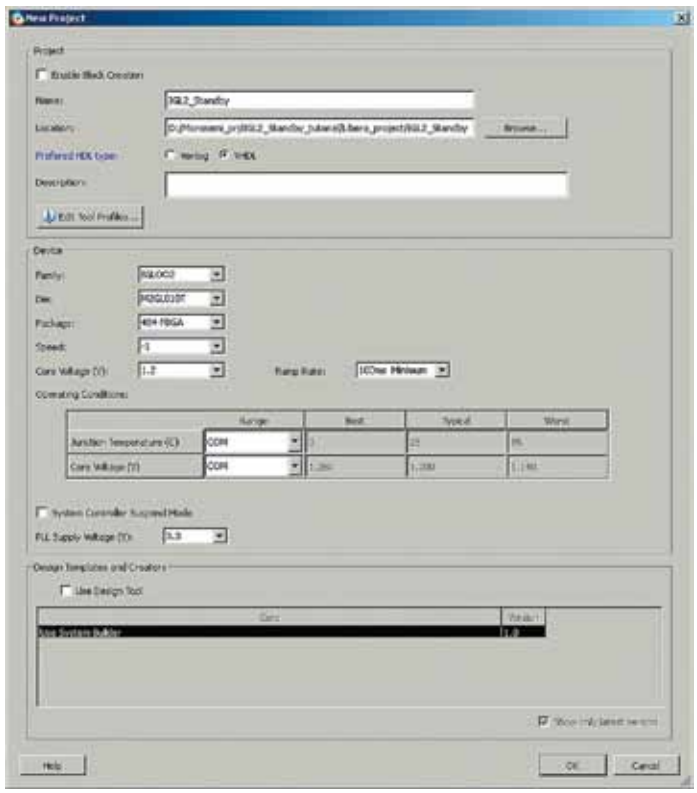

Рис. 2. Диалоговое окно создания нового проекта

Щелкнем по кнопке **OK**, окно создания нового проекта закроется.

Теперь посмотрим на главное окно **Libero SoC**. (Его описание можно найти в [6].) В его левой части увидим поле с закладками. Переключимся на закладку **Design Flow** (рис. 3). В ней отображается древовидная структура хода работы над проектом. Какие-то ветви дерева будут развернуты, какие-то свернуты. Сворачивать и разворачивать ветвь можно двойным щелчком мыши по ее имени или одиночным щелчком по маленькому квадратику рядом с именем.

Развернем ветвь Create Design. Найдем там строчку Create SmartDesign, щелкнем по ней правой кнопкой мыши и выберем из выпавшего меню пункт **Run** (рис. 4).

Появится небольшое диалоговое окно **Create New SmartDesign**. Введем в его единственное

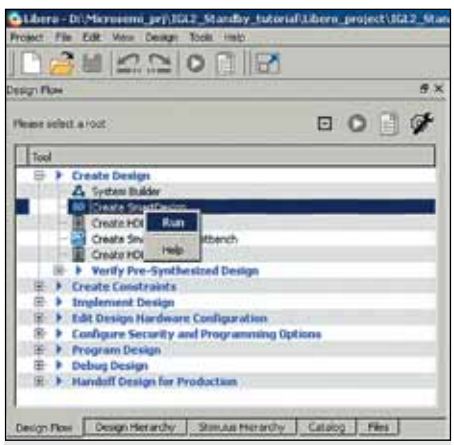

Рис. 3. Закладка Design Flow

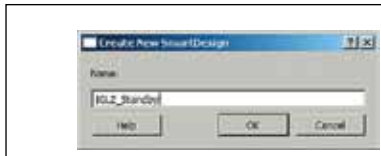

Рис. 4. Запуск создания проекта SmartDesign

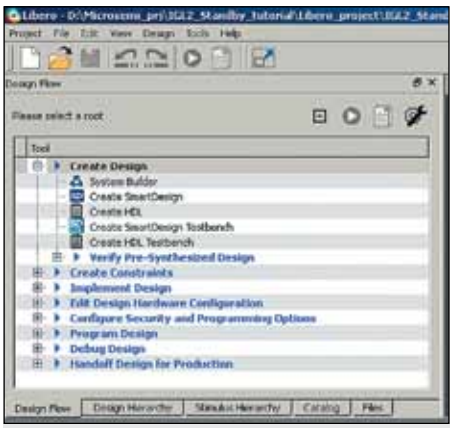

Рис. 5. Задание имени проекта SmartDesign

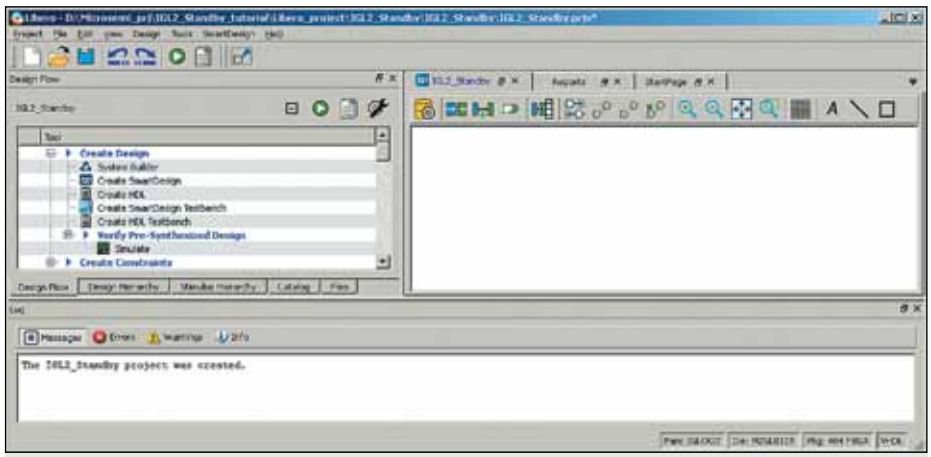

Рис. 6. Окно Libero SoC с пустым холстом нового проекта

поле **Name** имя нашего проекта IGL2_Standby и нажмем на кнопку **OK** (рис. 5).

После этого в правой части окна Libero SoC поверх ее остального содержимого появится поле интерактивного графического инструмента для создания проекта — «холст» (canvas) для нашего проекта (рис. 6).

Теперь мы будем собирать наш проект из готовых «кирпичиков» — IP-ядер из каталога Libero, а также модулей скачанного учебного проекта, используя интерактивный графический инструмент SmartDesign среды Libero SoC.

Начнем с формирователя тактовой частоты. Он должен генерировать тактовую частоту 100 МГц из входной частоты 32 кГц, поступающей от внешнего кварцевого генератора.

Переключимся в левой части окна Libero SoC с закладки **Design Flow** на закладку Catalog. Там мы также увидим древовидную структуру. Развернем ветвь Clock & Management (рис. 7).

Нам нужен блок Clock Conditioning Circuitry (CCC) v2.0.200. Мы видим, что в развернутой ветви Clock & Management он отображается бледно-серым курсивом. Это означает, что у нас на компьютере нет локальной копии самого блока. Проверим, что наше подключение к Интернету работает, щелкнем правой кнопкой мыши по имени интересующего нас блока и выберем из выпавшего меню пункт **Download** (рис. 8).

Через очень непродолжительное время имя блока в каталоге будет отображаться

Рис. 7. Выбор блока формирователя тактовой частоты из каталога

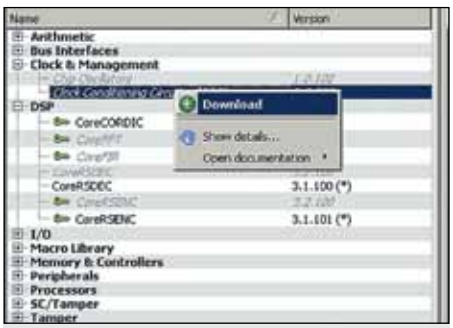

Рис. 8. Загрузка локальной копии компонента каталога

нормальным черным шрифтом, следовательно, локальная копия IP-ядра Clock Conditioning Circuitry успешно загрузилась. Перетащим мышью строчку с ее именем из каталога на наш холст. Здесь она перевоплотится в графический образ с именем FCCC_0 (рис. 9).

Теперь нужно выполнить конфигурацию нашего экземпляра IP-ядра. Для запуска конфигурации IP-ядра в среде Libero SoC 11.4 предусмотрено несколько способов:

- сделать двойной щелчок мышью по изображению IP-ядра на холсте;
- щелкнуть правой кнопкой мыши по этому изображению и выбрать из выпавшего меню пункт **Configure**;
- одиночный щелчок левой кнопкой мыши по гаечному ключу с шестеренкой в правом нижнем углу изображения IP-ядра.

В любом случае мы увидим окно конфигурации нашего IP-ядра (рис. 10).

Сделаем следующее:

- в поле **Reference Clock** впишем — 0.032 MHz;
- нажмем на кнопку под этим полем (не обращая внимания на текущее название этой кнопки) и пройдем по пунктам выпадающих меню **Oscillators → Crystal Oscillator**;
- установим флажок **GL0** и введем в поле **Frequency** справа от него число — 100;
- выберем в окне конфигурации закладку **Advanced**;
- щелкнем по второй сверху кнопке в левом верхнем углу окна (не обращая внимания на ее текущее название) и пройдем по пунктам выпадающих меню **Internal → PLL Internal** (рис. 11);
- перейдем к закладке **PLL Options** в окне конфигурации и установим флажок **Expose PLL_ARST_N and PLL_POWERDOWN_N signals** (рис. 12);

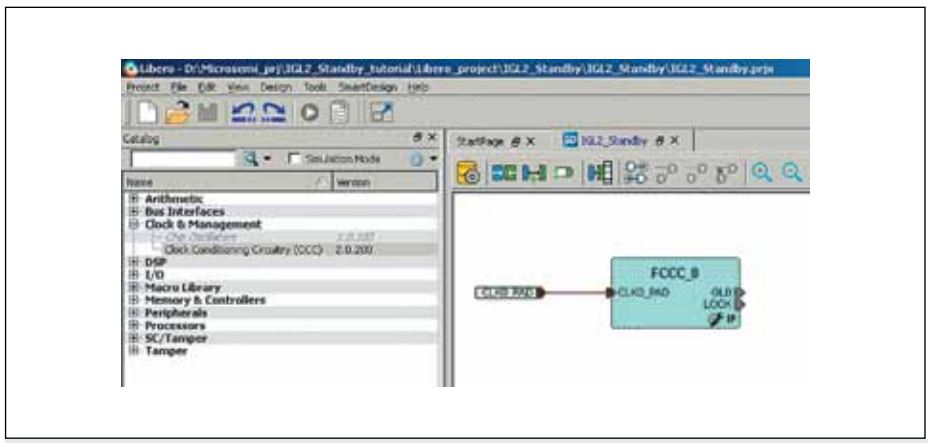

Рис. 9. Ввод в проект IP-ядра формирователя тактовой частоты

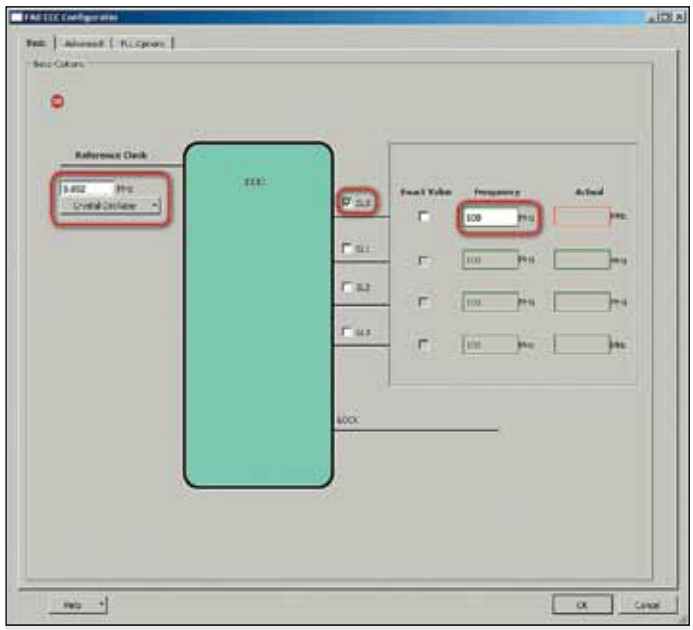

Рис. 10. Окно конфигурации IP-ядра формирователя тактовой частоты

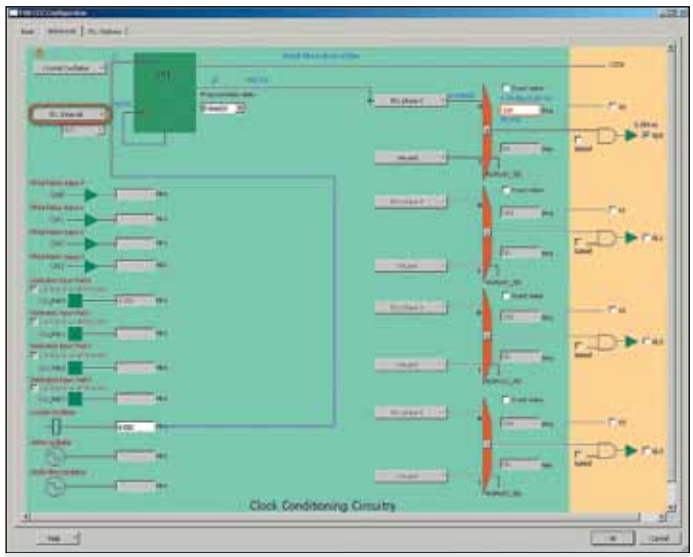

Рис. 11. Продолжение конфигурирования формирователя тактовой частоты

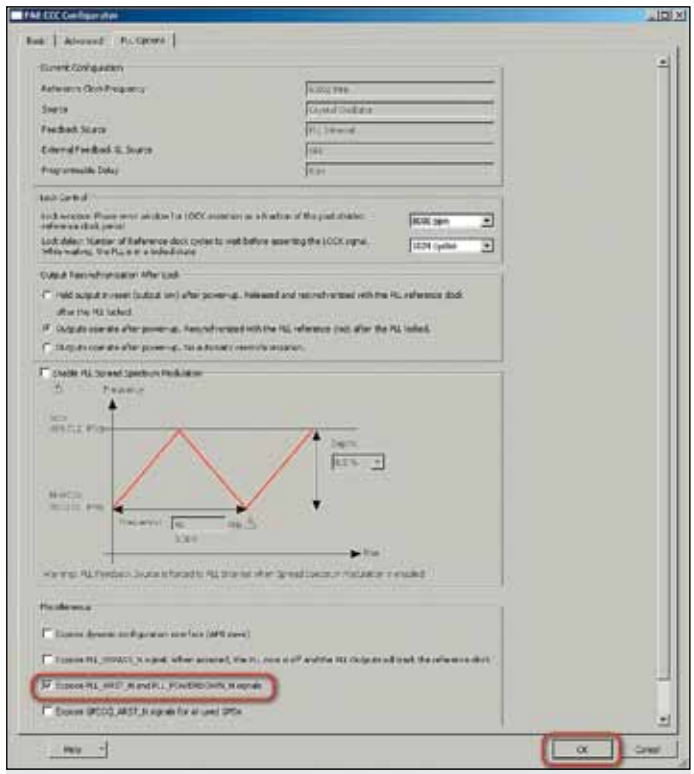

Рис. 12. Окончание конфигурирования формирователя тактовой частоты

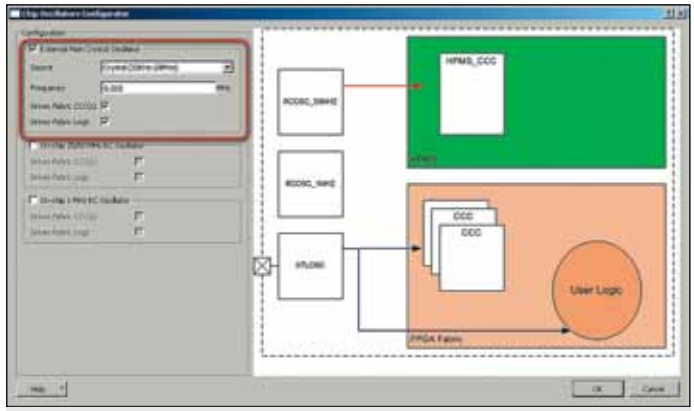

Рис. 13. Конфигурирование IP-ядра генератора

- щелкнем по кнопке **OK** и закроем окно конфигурирования формирователя тактовой частоты.

Затем поместим на наш холст IP-ядро генератора Chip Oscillators v1.0.102, загрузив при необходимости локальную копию, и сконфигурируем его (рис. 13):

- установим флажок **External Main Crystal Oscillator**;
- из выпадающего списка Source выберем вариант — Crystal (32 KHz - 20 MHz);
- в поле **Frequency** введем — 0.032;
- установим флажки **Drives Fabric CCC(s)** и **Drives Fabric Logic**;
- закроем окно конфигурации — щелкнем по кнопке **OK**.

Теперь добавим в наш проект блоки, написанные на языке VHDL. Они лежат в папке **Source_files** распакованного архива. Для этого надо сделать следующее:

- вернуться на закладку **Design Flow** в левой части окна Libero SoC;
- в ветви Create Design щелкнуть правой кнопкой мыши по строке Create HDL и выбрать из выпавшего меню пункт **Import Files…** (рис. 14);
- в открывшемся окне выбора файлов найти ту самую папку **Source_files** и войти в нее;
- одновременно нажать и отпустить клавиши **Ctrl+A** (латинское) — это команда «**Выбрать все**», после этого все файлы в диалоговом окне будут выделены;
- нажать на кнопку **Open** в правом нижнем углу диалогового окна.

Если мы теперь перейдем на закладку **Design Hierarchy** в левой части окна Libero SoC, то увидим там все добавленные файлы в правильном иерархическом порядке (рис. 15).

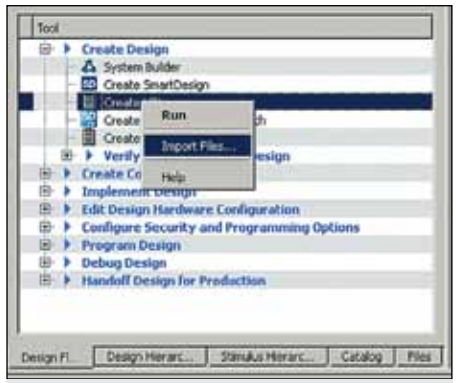

Рис. 14. Добавление в проект файлов на языке описания аппаратуры

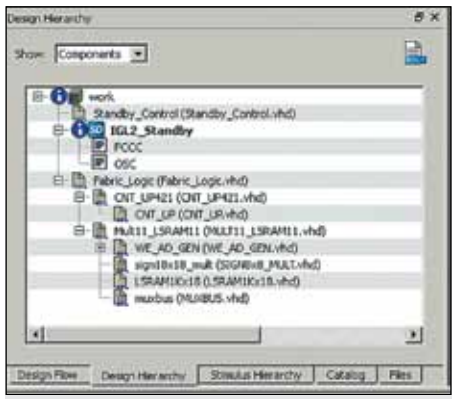

Рис. 15. Иерархия проекта

На верхнем уровне иерархии мы видим два модуля: Standby_Control и Fabric_Logic. Перетащим мышью строчки с их именами на холст. На холсте эти строчки, опять же, преобразятся в прямоугольные изображения модулей проекта. На данном этапе состав нашего проекта определился (рис. 16).

Теперь нам надо выполнить соединения компонентов на холсте. Перейдем в режим соединения, выбрав из меню Libero SoC пункт **SmartDesign — Connection Mode**. Вид указателя мыши при этом изменится. Соединим выход генератора с входом формирователя тактовой частоты следующим образом:

- щелкнем левой кнопкой мыши по порту XTLOSC_CCC_OUT компонента OSC_0 (щелкать можно не только по значку на границе изображения компонента, но и по имени порта), после чего порт окажется выделенным;
- нажмем клавишу **Ctrl** и, не отпуская ее, щелкнем по порту XTLOSC_CCC_IN компонента FCCC_0, теперь выделен и этот порт;
- щелкнем по одному из выделенных портов правой кнопкой мыши и выберем из выпавшего меню пункт **Connect**.

После этих действий на холсте возникнет соединение между портами.

Теперь по образу и подобию выполним соединения, перечисленные в таблице. Если надо соединить более двух портов, аналогично выделяем все эти порты при нажатой клавише **Ctrl**.

Таблица. Соединения на холсте

От	До
OSC_0: XTLOSC_O2F	Standby_Control_0: CLK
Standby_Control_0: PLL_PowerDown	FCCC_0: PLL_ARST_N FCCC_0: PLL_POWERDOWN_N
FCCC_0: GL0	Fabric_Logic_0: CLK
FCCC_0: LOCK	Fabric_Logic_0: RST

Далее надо вывести часть сигналов на выводы ПЛИС. В терминологии среды Libero SoC это называется Promote to Top Level. Чтобы вывести сигнал на выводы ПЛИС, надо щелкнуть по его порту правой кнопкой мыши и выбрать из выпавшего меню пункт

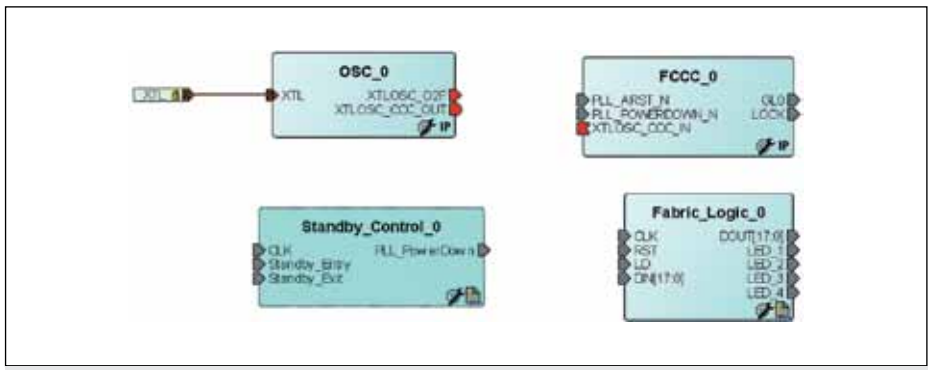

Рис. 16. Вид холста после добавления модулей проекта

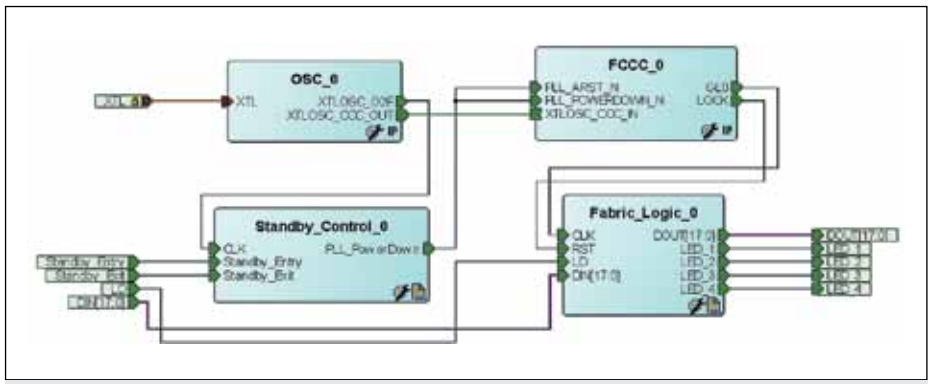

Рис. 17. Вид холста после выполнения соединений

Promote to Top Level. Выведем наружу следующие порты:

- Standby_Control_0: Standby_Entry;
- Standby_Control_0: Standby_Exit;
- Fabric_Logic_0: LD;
- Fabric_Logic_0: DIN [17:0];
- Fabric_Logic_0: DOUT [17:0];
- Fabric_Logic_0: LED_1;
- Fabric_Logic_0: LED_2;
- Fabric_Logic_0: LED_3;
- Fabric_Logic_0: LED_4.

Теперь все соединения сделаны (рис. 17). Для улучшения читаемости схемы можно перетащить компоненты в более удобные места мышью и/или доверить это среде Libero SoC: щелкнуть правой кнопкой мыши в любом свободном месте холста и выбрать из выпавшего меню пункт **Auto Arrange Instances**.

Сохраним наш проект, нажав на панели инструментов окна Libero SoC кнопку с изображением дискеты.

Следующий шаг — генерация компонента SmartDesign. Выбираем пункт меню главного окна **SmartDesign — Generate Component** и дожидаемся появления в поле протокола в нижней части главного окна сообщения **Info: 'IGL2_Standby' was successfully generated**.

Импорт файла физических ограничений

Сейчас нам надо привязать наш проект к реальной микросхеме, установленной на плате отладочного набора, а именно: указать соответствие выведенных наружу сигналов проекта конкретным выводам микросхемы и для каждого вывода задать его атрибуты — направление передачи сигнала и прочее. Такая привязка задается в файле физических ограничений (Physical Constraint file). В среде Libero SoC эти файлы имеют расширение PDC. В папке, которую мы получили в начале работы из архива, есть подпапка Constraints, а в ней лежит тот самый нужный нам файл *IGL2_Standby.pdc*. Импортировать в проект его можно следующим образом:

- перейти на закладку Design Flow в левой части окна Libero SoC;
- развернуть ветвь Create Constraints;
- выбрать в этой ветви строку I/O Constraints, щелкнуть по ней правой кнопкой мыши и выбрать из выпавшего меню

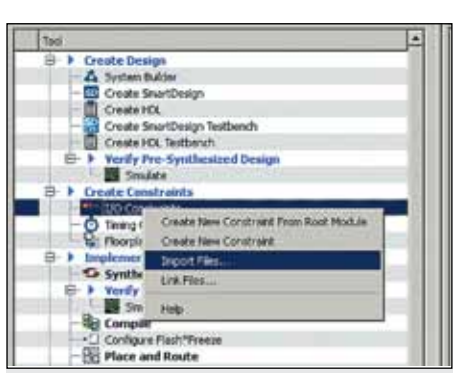

Рис. 18. Импорт файла физических ограничений

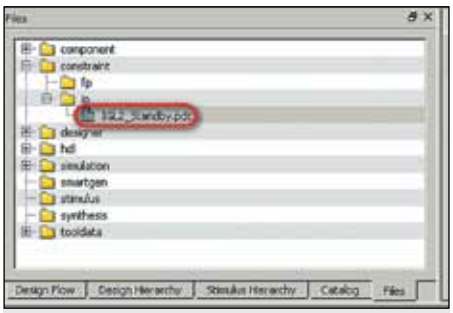

Рис. 19. Результат импорта файла физических ограничений

пункт **Import Files** (рис. 18). Появится обычное диалоговое окно выбора файла. Найдем через него наш файл *.pdc* и нажмем **OK**. В появившемся окне Information в ответ на вопрос **Do you want to organize constraint file(s) for your current root (IGL2_Standby) for (Compile)** надо нажать на кнопку **No**.

Теперь этот файл можно найти на закладке **Files** в левой части окна Libero SoC в разделе **Constraint — io** (рис. 19).

Описание построения файла физических ограничений можно найти в справочной системе Libero SoC (Help — Help Topics — Implement Design — Constrain Place and Route — Assigning Design Constraints — Design Constraints Guide — Reference — Constraints by File Format — PDC Command Reference).

Синтез и трассировка

Следующий этап работы — синтез и трассировка проекта. Для нас это будет последовательностью несложных действий.

1) В левой части окна Libero SoC на закладке **Design Flow** развернем ветвь Create Constraints, в ней развернем ветвь I/O Constraints и увидим имя нашего файла физических ограничений. Оно помечено перечеркнутым красным кругом. Щелкнем по нему правой кнопкой мыши и выберем из выпавшего меню пункт **Use for Compile**. После этого перечеркнутый красный круг сменится на зеленую галочку.
2) Запустим генерацию данных для программирования и выберем из меню Libero SoC пункт **Design — Generate Bitstream** (в других версиях среды возможно название Generate Programming Data) или найдем в верхней части закладки Design Flow кнопку с зеленым кругом и белым треугольником внутри и щелкнем по ней мышью. Дальнейшая деятельность Libero SoC пойдет в пакетном режиме, среда будет самостоятельно запускать нужные программы в соответствующем порядке. Нам останется только наблюдать за сообщениями в поле протокола в нижней части окна Libero SoC.

Если в проекте нет ошибок, то по мере выполнения этапов работы можно увидеть в разделе **Implement Design** закладки **Design Flow** появление галочек рядом с именами этих этапов. Причем сама закладка **Design Flow** будет неактивна, за исключением красной кнопки **Abort**, которая позволит при необходимости прервать процесс. Мы узнаем об окончании процесса по переходу закладки **Design Flow** в активное состояние и по сообщению **Generating Bitstream File Finished** в поле протокола.

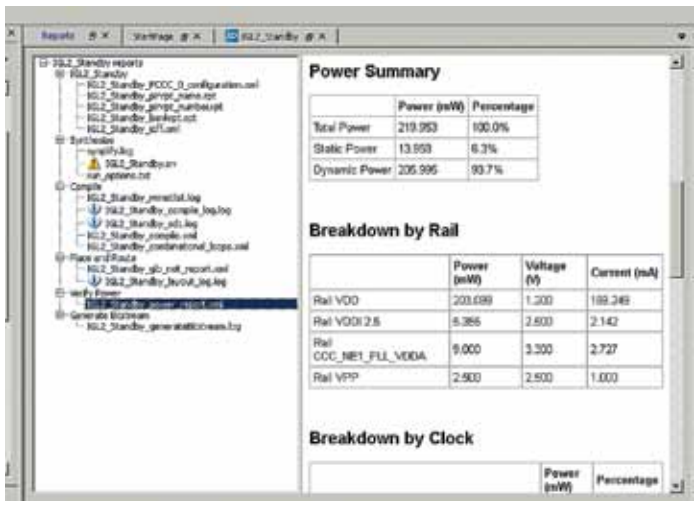

Рис. 20. Отчет о потребляемой мощности

Генерация отчета о потреблении

После окончания трассировки мы сможем увидеть один из основных результатов нашей лабораторной работы — отчет о потреблении (Power Report). Найдем в разделе **Implement Design** закладки **Design Flow** ветвь Verify Post Layout Implementation, развернем ее, если она свернута, и щелкнем правой кнопкой мыши по строке Verify Power, а затем выберем из выпавшего меню пункт **Run**. Через некоторое время в правой верхней части окна Libero SoC активизируется закладка **Reports**, а на ней мы увидим подробный отчет о расчетной потребляемой мощности (рис. 20): суммарно, статическая, динамическая, по отдельным напряжениям питания и т. д. Отметим, что для нашего объемистого проекта предсказано суммарное потребление всего в 220 мВт.

Программирование

И наконец, пришло время взять в руки отладочную плату из набора IGLOO2 Evaluation Kit (рис. 21).

Для того чтобы запрограммировать ПЛИС, нам надо сделать следующее.

1) До начала программирования (и до включения питания) установить джамперы J3

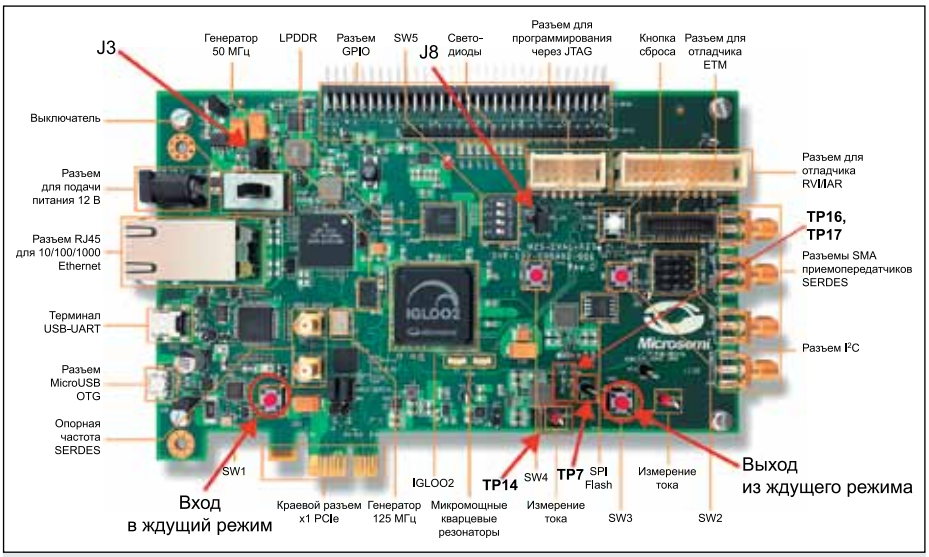

Рис. 21. Отладочная плата из набора IGLOO2 Evaluation Kit

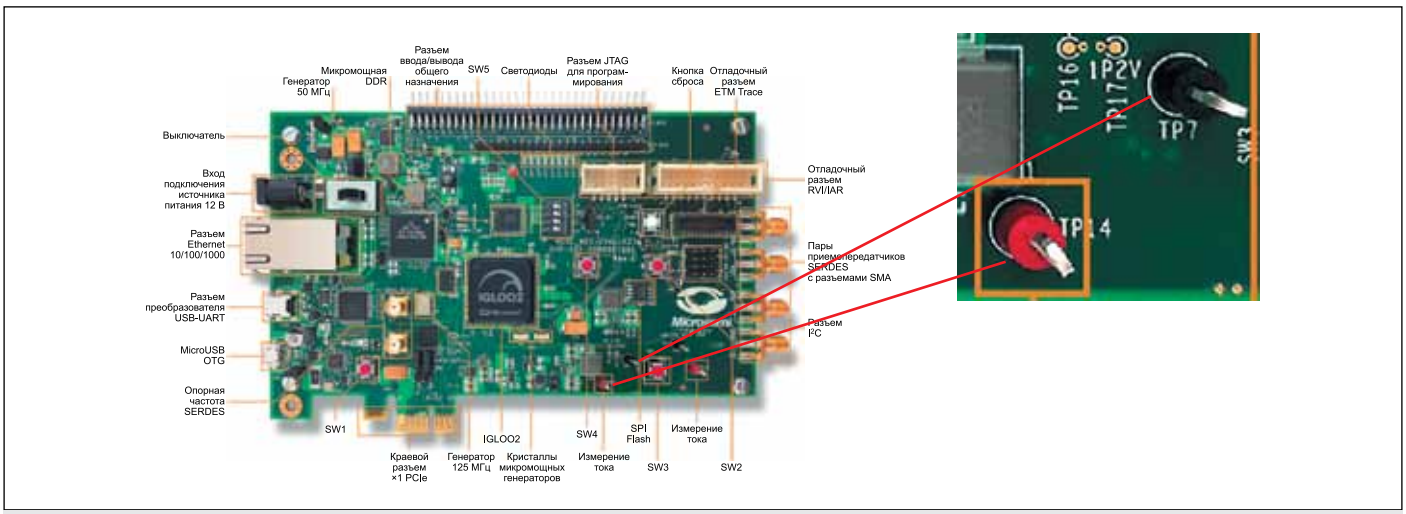

Рис. 22. Расположение точек для измерения на отладочной плате

и J8 так, как показано на рисунке (в самое нижнее (по рисунку) из возможных положений).

2) Подключить ленточный кабель программатора FlashPro4 к разъему J5 (разъем для программирования через JTAG).
3) Соединить FlashPro4 с портом USB компьютера при помощи кабеля miniUSB.
4) Если появится запрос установки драйверов — установить драйверы FlashPro4. Они находятся в папке установки Libero SoC, в подпапке \Designer\Drivers.
5) Подать питание на отладочную плату. Для этого надо подключить к плате сетевой адаптер и передвинуть выключатель в левое (по рисунку) положение. После этого должны загореться три зеленых светодиода в левом верхнем углу платы.
6) Развернуть ветвь Program Design в нижней части закладки **Design Flow**, щелкнуть правой кнопкой мыши по строчке Run PROGRAM Action и выбрать из выпавшего меню пункт **Run**. Утилита управления программатором запустится в пакетном режиме, ее работу можно отслеживать в поле протокола. На время программирования закладка **Design Flow** также станет неактивной. По окончании программирования в поле протокола среди прочих сообщений появится сообщение **Chain Programming Finished**. Об успешном окончании программирования будет также свидетельствовать зеленая галочка возле строки PROGRAM Action на закладке **Design Flow**. Отметим, что прерывать программирование нельзя — это может привести к повреждению микросхемы или программатора.

Если после окончания программирования мы посмотрим на отладочную плату, то увидим, что правые четыре светодиода в группе из восьми светодиодов левее разъема JTAG (E1, F4, F3 и G7) начали переключаться с различной частотой.

Теперь окно Libero SoC можно закрыть.

Измерение потребляемой мощности

На отладочной плате, как уже упоминалось, есть специальная схема для измерения тока, потребляемого ядром ПЛИС. Эта схема усиливает падение напряжения на специальном токоизмерительном резисторе сопротивлением 0,05 Ом, установленном в цепи питания ядра. Выходное напряжение этой схемы надо измерять между точками TP7 (отрицательный щуп вольтметра) и TP14 (положительный щуп) (рис. 22).

Если полученное напряжение в милливольтах разделить на пять, получится ток потребления в миллиамперах. Затем умножим этот ток на напряжение питания ядра 1,2 В и получим потребляемую мощность в милливаттах.

Кроме того, можно измерить напряжение непосредственно на токоизмерительном резисторе. Для этого надо подключить вольтметр к точкам TP17 (отрицательный щуп) и TP16 (положительный щуп). Если измеренное напряжение в милливольтах поделить на 0,05 и умножить на 1,2, то мы получим потребляемую ядром ПЛИС мощность в милливаттах. Заметим, что при ожидаемом токе менее 10 мА падение напряжения составит меньше 0,5 мВ, поэтому для таких измерений нужно использовать точный вольтметр с субмилливольтовым диапазоном измерения.

Мы начнем с измерения полной потребляемой мощности. Подадим питание на отладочную плату. Убедимся, что светодиоды E1, F4, F3 и G7 (рис. 21) переключаются. Если это не так (что очень маловероятно), то нажмем кнопку сброса. Измерим напряжение между точками TP7 и TP14. У авторов получилось 865 мВ. Это соответствует потребляемой мощности 207,6 мВт. Полученный результат с очень высокой точностью совпадает с величиной полного потребления 205,995 мВт (рис. 20), предсказанной в отчете по потребляемой мощности.

Теперь нажмем кнопку SW1 и переведем ПЛИС в ждущий режим. При этом переключение светодиодов E1, F4, F3 и G7 должно остановиться. Снова измерим напряжение между точками TP7 и TP14. У авторов получилось 60,2 мВ, или 14,4 мВт. Это достаточно хорошо совпадает с величиной статического потребления 13,958 мВт из отчета по потребляемой мощности.

Заключение

В нашей лабораторной работе мы провели измерение потребляемой мощности ПЛИС семейства IGLOO2 корпорации Microsemi в рабочем режиме и в ждущем режиме. Выяснилось, что:

- в ждущем режиме потребляемая мощность, и так небольшая, снизилась более чем в 10 раз;
- САПР Libero SoC позволяет с высокой точностью оценить мощность потребления для откомпилированного проекта.

Кроме того, мы приобрели начальные сведения и навыки по работе в САПР Libero SoC версии 11.4.

Литература

1. IGLOO2 FPGA Low Standby Power — Libero SoC v11.4 Demo Guide. Материал с сайта корпорации Microsemi.
2. Иоффе Д., Казаков А. SmartFusion2 и IGLOO2 — надежные, экономичные, компактные. Обзор новых семейств ПЛИС корпорации Microsemi // Компоненты и технологии. 2014. № 8.
3. Иоффе Д., Казаков А. SmartFusion2 и IGLOO2 — в помощь разработчику. Обзор отладочных плат для новых семейств ПЛИС корпорации Microsemi // Компоненты и технологии. 2014. № 10.
4. IGLOO2 FPGA Low Power Design User Guide. Материал с сайта корпорации Microsemi.
5. www.microsemi.com
6. Иоффе Д. Libero SoC — быстрый старт // Компоненты и технологии. 2013. № 10.