Блоки SERDES в новых ПЛИС корпорации Microsemi. Часть 2

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

Этот материал продолжает учебный курс, который посвящен блокам SERDES, встроенным в новые семейства ПЛИС корпорации Microsemi.

В первой части статьи [1] мы познакомились с кратким описанием блоков SERDES, встроенных в новые семейства ПЛИС корпорации Microsemi — IGLOO2 и SmartFusion2, а также подготовились к лабораторной работе по исследованию SERDES: установили необходимое программное обеспечение и проверили работу оборудования.

Сегодня мы освоим на практике создание проекта с использованием блоков SERDES в среде Libero SoC корпорации Microsemi и будем работать с отладочным набором IGLOO2 Evaluation Kit. Процедура работы с набором SmartFusion2 Evaluation Kit очень похожа, различия мы будем оговаривать во время самого процесса.

Шаг 1. Создание проекта в среде Libero SoC

Запускаем САПР Libero SoC версии 11.4 или более поздней.

Выбираем из меню главного окна пункт Project > New Project. Появится окно создания нового проекта (рис. 1).

Заполним его так, как показано на рисунке.

В группе Project:

- не будем устанавливать флажок Enable Block Creation;
- в поле Name введем имя проекта, например EPCS_Demo;
- нажмем на кнопку Browse… рядом с полем Location и укажем папку, в которой будут располагаться файлы нашего проекта, например D:/Projects/Microsemi;
- выберем Verilog в качестве предпочтительного языка описания аппаратуры (Preferred HDL type).

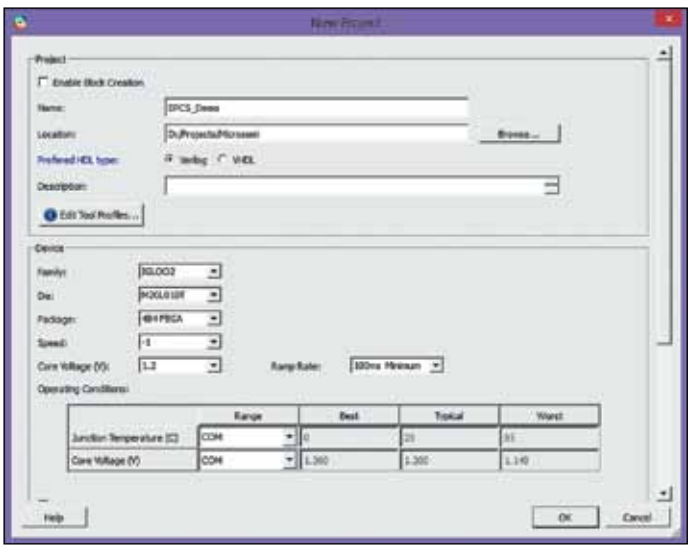

Рис. 1. Окно New Project

В группе Device:

- из выпадающего списка Family выберем, в зависимости от того, какой у нас отладочный набор, семейство IGLOO2 или SmartFusion2;
- из выпадающего списка Die выберем кристалл M2GL010T для семейства IGLOO2 или M2S090T для SmartFusion2;
- тип корпуса (Package): 484FBGA;
- градацию скорости (Speed): –1;
- напряжение питания ядра (Core Voltage): 1,2;
- скорость нарастания напряжения питания (Ramp Rate, только для семейства SmartFusion2): 100 ms Minimum.
- рабочие условия (Operating Conditions): во всей таблице выберем вариант COM («коммерческий» диапазон температур).

Теперь обратим внимание на полосу прокрутки в правой части окна New Project. Протянем скроллер вниз, чтобы увидеть оставшиеся управляющие элементы окна (рис. 2), и продолжим заполнение.

- флажок System Controller Suspend Mode оставим сброшенным;
- напряжение питания ФАПЧ (PLL Supply Voltage): 3.3.

Теперь вернемся к верхней части окна New Project и нажмем кнопку Edit Tool Profiles. Появится окно Tool Profiles (рис. 3), в котором мы смо-

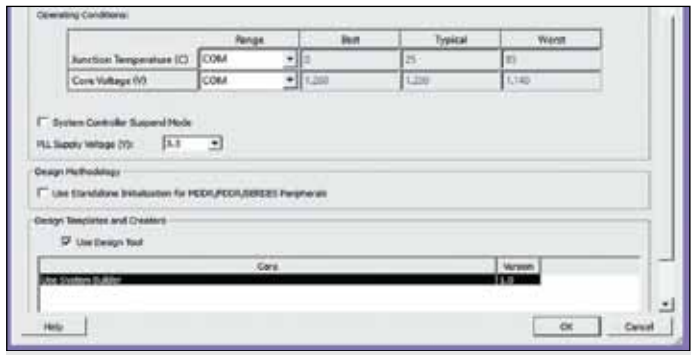

Рис. 2. Нижняя часть окна New Project

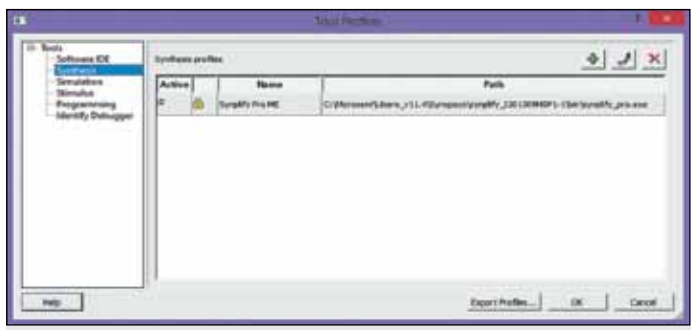

Рис. 3. Окно Tool Profiles

Рис. 4. Диалоговое окно System Builder

жем проконтролировать правильность выбора компонентов среды Libero SoC. Для этого надо будет выбирать имя компонента в левой части окна и контролировать его настройку в правой части.

Убедимся, что:

- в качестве синтезатора (Synthesis в левой части окна) указан Synplify Pro ME I-2013.09M-SP1;
- для моделирования (Simulation) выбран ModelSim ME;
- для программирования ПЛИС (Programming) указан FlashPro.

Теперь нажмем на кнопку OK и закроем окно управления профилями инструментов, а затем кнопкой OK закроем окно New Project. После этого появится диалоговое окно System Builder (рис. 4). При необходимости это окно можно впоследствии вызвать из окна Design Flow через пункт System Builder в разделе Create Design.

Введем в качестве имени системы EPCS_Demo и нажмем OK. Появится окно инструмента System Builder (рис. 5).

Это окно содержит несколько разделов-страниц. Для переключения между ними служат кнопки Next и Back, расположенные в правом нижнем углу окна. Название текущего раздела можно определить по цвету стрелок в верхней части окна.

Итак, перед нами первый раздел, Device Features. Стрелка с надписью Device Features имеет ярко-зеленый цвет, остальные стрелки — серые. Установим флажок SERDESIF_0, затем флажок HPMS On-chip Flash Memory (eNVM). Заметим, что с каждой установкой флажка в окне System Builder происходят некие изменения, которые полезно отслеживать.

Щелкнем по кнопке Next два раза, перейдя, таким образом, в раздел Peripherals (рис. 6). После этого ярко-зелеными станут все стрелки от Device Features до Peripherals.

Если мы работаем с IGLOO2, то оставляем в неприкосновенности все значения, предложенные по умолчанию. Если же у нас устройство SmartFusion2, устанавливаем настройки так, как указано на рис. 6.

Щелкаем еще раз по кнопке Next и попадаем в раздел Clocks. Здесь устанавливаем следующие параметры.

Для IGLOO2:

- в группе System Clock вводим тактовую частоту 125 МГц и выбираем из выпадающего списка пункт FPGA Fabric Input;
- в поле HPMS_CLK вводим 100.00 МГц;
- в поле FIC_0_CLK выбираем из выпадающего списка число 2.

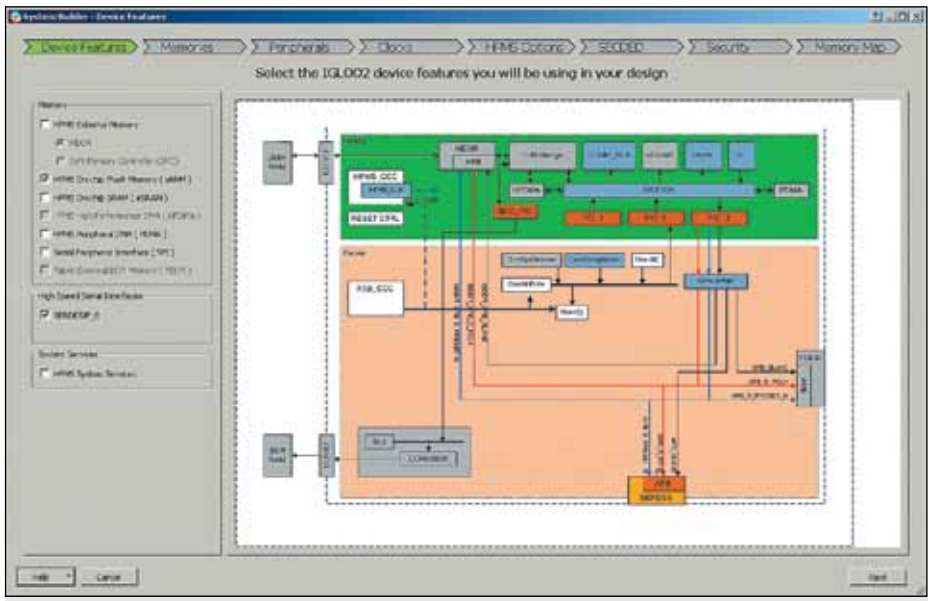

Рис. 5. Окно System Builder

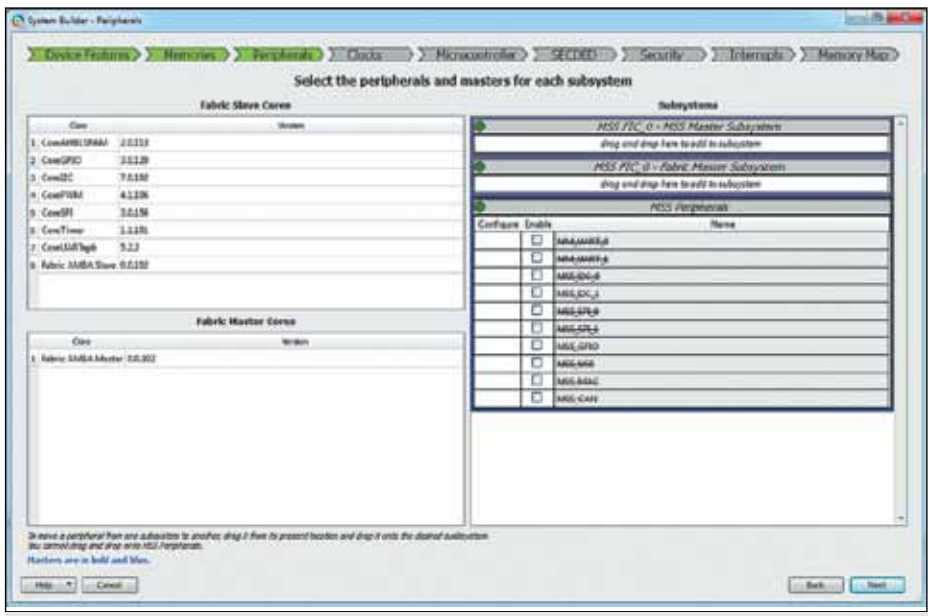

Рис. 6. Настройки раздела Peripherals окна System Builder для отладочного набора на базе устройства SmartFusion2

Для SmartFusion2:

- в группе System Clock также вводим тактовую частоту 125 МГц и выбираем из выпадающего списка пункт FPGA Fabric Input;
- в поле M3_CLK вводим 100.00 МГц;
- для параметров APB_1_CLK и APB_2_CLK выбираем из выпадающего списка делитель 1;
- для параметра FIC_0_CLK выбираем из выпадающего списка делитель 2;
- перейдем на закладку Fabric CCC, установим флажок FAB_CCC_GL1 и введем в поле рядом с этим флажком 50.00.

Далее щелкаем по кнопке Next, пока не доберемся до последнего раздела Memory Map. Во всех разделах оставляем значения по умолчанию. По окончании этой процедуры System Builder потратит некоторое время

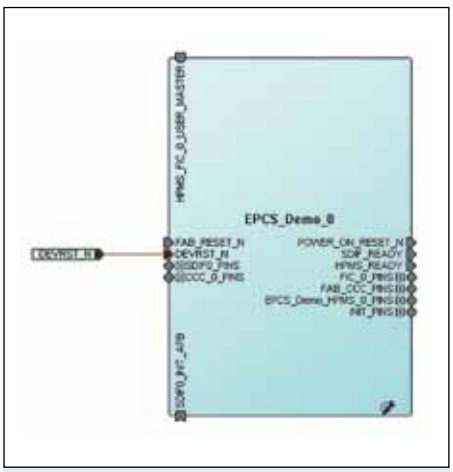

Рис. 7. Компонент System Builder для IGLOO2

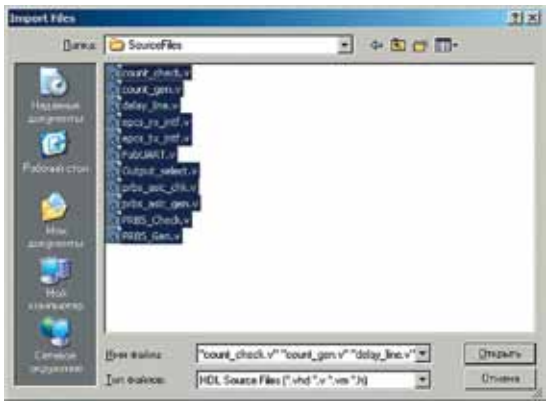

Рис. 8. Выбор файлов с исходными текстами пользовательской логики

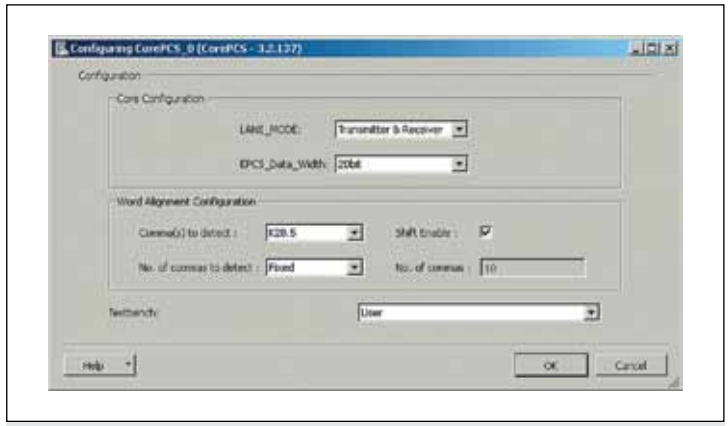

Рис. 11. Настройка ядра CorePCS

на компоновку системы, показывая нам при этом небольшое окно с ползущим индикатором. Теперь можно нажимать на кнопку Finish, которая появилась в правом нижнем углу окна System Builder.

Подождем еще немного. Помним, что на запросы брандмауэра, если таковые появятся, необходимо разрешать подключение. После этого блок, созданный при помощи инструмента System Builder, появится на отдельной вкладке в главном окне среды Libero SoC (рис. 7).

Рядом с некоторыми именами в поле компонента есть квадратики с плюсами внутри. Это означает, что данное имя описывает не один вывод компонента, а свернутую группу выводов. Если щелкнуть мышью по плюсу внутри квадратика, то группа развернется и можно увидеть входящие в нее элементы.

Если мы работаем с семейством IGLOO2, то System Builder автоматически подключит два IP-ядра, CoreResetP и CoreConfigP. Они обеспечивают сброс и конфигурацию периферийных устройств. В нашем проекте они будут выполнять сброс и конфигурацию модуля SERDESIF. Этот модуль содержится в компоненте, который мы создали при помощи System Builder.

Шаг 2. Импорт пользовательской логики в проект

Исходные тексты необходимой для нашего проекта пользовательской логики на языке описания аппаратуры Verilog находятся в распакованном содержимом нашего архива, в папке SourceFiles. Чтобы импортировать эту логику в проект, надо выполнить следующие действия.

1. В главном окне среды Libero SoC выбрать пункт меню File -> Import -> HDL Source Files.
2. В появившемся диалоговом окне найти указанную папку.
3. Выбрать все файлы в папке (рис. 8), обведя их мышью или нажав на клавиатуре одновременно клавиши Ctrl и A (латинское), и нажать на кнопку «Открыть» (Open).

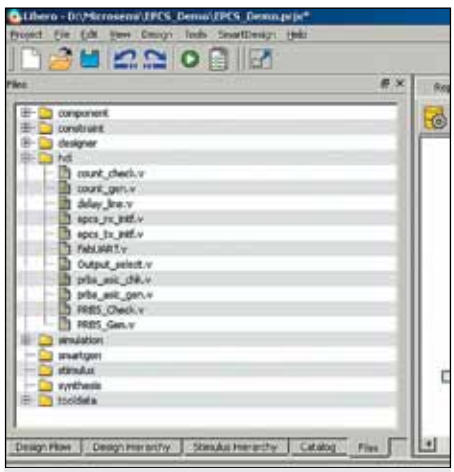

Рис. 9. Представление файлов пользовательской логики на закладке Files

Теперь можно увидеть эти файлы в левой части окна Libero SoC на закладке Files (рис. 9).

Шаг 3. Сборка компонентов из каталога Libero SoC при помощи SmartDesign

Сейчас в правой верхней части главного окна Libero SoC должна быть активна закладка EPCS_Demo_top. Если это не так, активизируем ее. Перейдем на закладку Catalog в левой верхней части главного окна. Развернем в каталоге раздел Peripherals, выберем IP-ядро COREUART и перетащим его мышью на холст EPCS_Demo_top. На холсте появится экземпляр (instance) этого ядра COREUART_0. Сделаем двойной щелчок мышью по нему и зададим его параметры так, как показывает рис. 10, и нажмем кнопку OK.

Теперь перетащим на холст из того же раздела каталога Peripherals модуль CorePCS, сделаем на его экземпляре двойной щелчок мышью и установим в окне настройки следующие параметры (рис. 11).

Далее найдем в каталоге в разделе Macro Library модуль AND2 и создадим на холсте два экземпляра этого модуля.

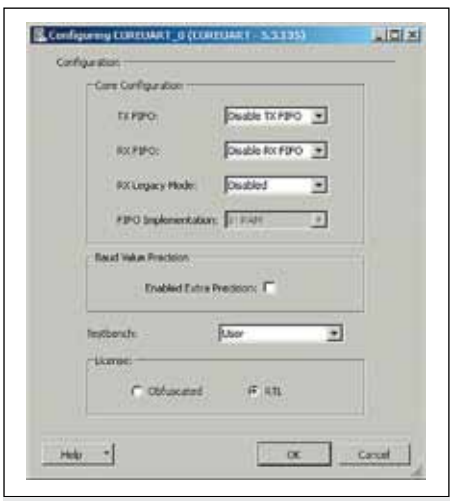

Рис. 10. Установка параметров ядра COREUART

Шаг 4. Создание иерархии проекта SmartDesign и добавление компонента SERDESIF

Выберем в левой верхней части окна Libero SoC закладку Design Flow, развернем в ней раздел Create Design и дважды щелкнем в нем по строке Create SmartDesign. Появится диалоговое окно Create New SmartDesign. Введем в единственное поле этого окна Name имя вновь создаваемого модуля SmartDesign: EPCS_SERDES. Затем в правой верхней части окна Libero SoC появится новая закладка с холстом под этим именем. Теперь мы можем создавать модуль проекта при помощи интерактивного инструмента SmartDesign.

Прежде чем начать работу с каталогом, сделаем небольшое замечание. Окно каталога отображает содержимое хранилища (vault), в котором хранятся готовые IP-ядра. Часть этих ядер появляется в хранилище сразу после установки среды Libero SoC, остальные ядра скачиваются в разное время по мере необходимости из интернет-репозитория корпорации Microsemi. В ходе работы с каталогом могут возникнуть непредвиденные сложности, например из-за конфликтов

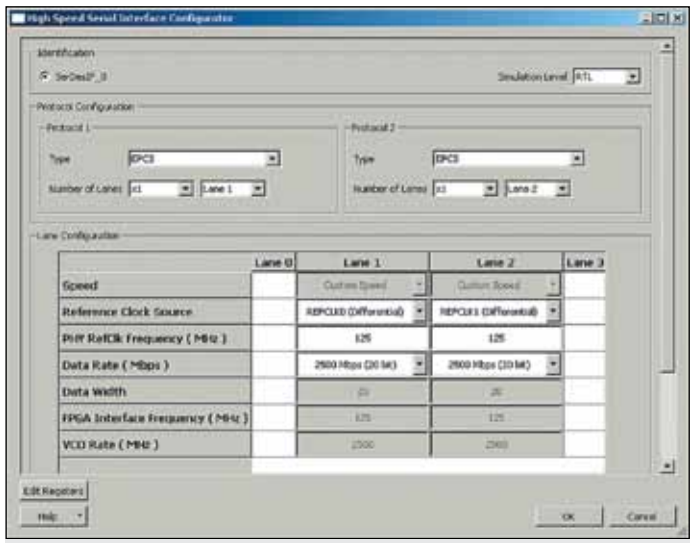

Рис. 12. Конфигурирование IP-ядра High Speed Serial Interface

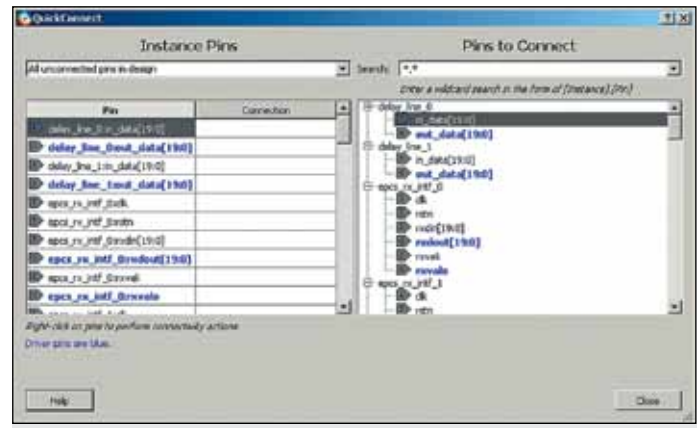

Рис. 13. Окно Quick Connect

версий IP-ядер. На этот случай Microsemi предоставляет на своем сайте автономные полные версии репозитория для разных версий Libero SoC. Они выложены в виде ZIP-архивов. При необходимости можно найти страницу с этими архивами в разделе скачивания (download) для Libero SoC по ссылке Download Standalone Vault. Такой архив надо распаковать в специально созданную папку и указать среде Libero SoC путь к ней. Для этого надо выбрать пункт меню Project — Vault/Repositories Settings. Появится одноименное диалоговое окно. В его левом поле следует выбрать пункт Vault Location, затем нажать на кнопку с многоточием рядом с полем Select New Vault Location и через появившееся окно «Проводника» найти папку с распакованным автономным репозиторием.

Итак, откроем каталог, развернем раздел Peripherals и перетащим оттуда на холст SmartDesign модуль High Speed Serial Interface. Сделаем двойной щелчок мышью по экземпляру на холсте. Появится окно конфигуратора (рис. 12).

Установим параметры IP-ядра так, как показано на этом рисунке.

В группе Identification:

- SerDesIF_0;
- SimulationLevel: RTL.

В группе Protocol Configuration:

- Protocol 1: EPCS, x1, Lane 1;
- Protocol 2: EPCS, x1, Lane 2.

В группе Lane Configuration:

- Speed: для обоих каналов CUSTOM SPEED;
- Reference Clock Source: REFCLK1 (Differential);
- PHY RefClk Frequency (MHz): 125;
- Data Rate (Mbps): 2500 Mbps (20 bit) for all Lanes.

Затем щелкнем по кнопке OK, чтобы применить сделанные настройки и закрыть окно конфигуратора.

Теперь нам нужно вставить в проект импортированные модули, созданные на языке описания аппаратуры. Откроем закладку Design Hierarchy в левой верхней части окна Libero SoC и найдем на ней следующие модули: epcs_tx_intf, epcs_rx_intf и delay_line. Добавим на холст по два экземпляра каждого из этих модулей.

Итак, все необходимые нам компоненты лежат на холсте. Соединим их между собой. SmartDesign предлагает три метода создания связей.

Первый метод: в окне SmartDesign щелкаем правой кнопкой мыши в любом свободном месте на холсте и выбираем из выпавшего меню пункт Connection Mode. Форма указателя мыши после этого изменится. Далее выбираем первый из выводов создаваемого соединения и перетаскиваем указатель мыши на второй вывод этого соединения.

Второй метод: нажимаем на клавиатуре клавишу Ctrl и, не отпуская ее, щелкаем левой кнопкой мыши по всем выводам, подлежащим соединению. Затем щелкаем правой кнопкой мыши по одному из выбранных выводов и выбираем из выпавшего меню пункт Connect.

И третий метод: щелкаем правой кнопкой мыши в любом свободном месте на холсте и выбираем из выпавшего меню пункт Quick Connect. Появится окно Quick Connect (рис. 13).

Выберем в столбце Instance Pin вывод, который мы будем подключать, затем в столбце Pins to Connect — другой вывод, с которым мы хотим соединить первый вывод, щелкнем правой кнопкой по одному из выбранных выводов и выберем из выпавшего меню пункт Connect. Аналогично можно ликвидировать ненужное соединение, выбрав из этого выпавшего меню пункт Disconnect. Если мы щелкаем правой кнопкой мыши не по одиночному выводу, а по шине, то в выпадающем меню появляется еще один пункт — Edit Slice. Он позволяет разбивать шину на отдельные группы и редактировать их. И наконец, если щелчок правой кнопкой мыши пришелся на имя в столбце Instance Pins, то в выпавшем меню будут еще и пункты Promote to Top Level, Tie Low, Tie High, Mark Unused и другие, позволяющие подключать выводы к заданным логическим уровням, к выводам ПЛИС и прочее. Таким образом, инструмент Quick Connect предоставляет нам множество возможностей для редактирования соединений в удобной форме.

Любым из этих методов соединим между собой выводы компонентов проекта (табл. 1). Если мы выбрали первый или второй метод, то помним, что некоторые выводы могут быть спрятаны внутри свернутых шин и перед выполнением соединения шины надо развернуть.

Теперь выведем часть сигналов нашего модуля для подключения верхнего уровня, например для использования этого модуля как компонента более высокого уровня или для соединения данных сигналов с выводами корпуса ПЛИС. Для этого надо щелкнуть правой кнопкой мыши по имени сигнала и выбрать из выпавшего меню пункт Promote to top level. Иногда для улучшения читаемости схемы

Таблица 1. Соединения между компонентами проекта

От	До
SERDES_IF_0:EPCS_1_TX_DATA[19:0]	epcs_tx_intf_1:txdout[19:0]
SERDES_IF_0:EPCS_1_RX_VAL	epcs_rx_intf_1:rxvali
SERDES_IF_0:EPCS_1_RX_DATA[19:0]	delay_line1:in_data[19:0]
SERDES_IF_0:EPCS_1_RX_RESET_N	epcs_rx_intf_1:rstn
SERDES_IF_0:EPCS_1_TX_RESET_N	epcs_tx_intf_1:rstn
SERDES_IF_0:EPCS_1_RX_CLK	epcs_rx_intf_1:clk
SERDES_IF_0:EPCS_1_TX_CLK	epcs_tx_intf_1:clk
SERDES_IF_0:EPCS_2_TX_DATA[19:0]	epcs_tx_intf_2:txdout[19:0]
SERDES_IF_0:EPCS_2_RX_VAL	epcs_rx_intf_2:rxvali
SERDES_IF_0:EPCS_2_RX_DATA[19:0]	delay_line2:in_data[19:0]
SERDES_IF_0:EPCS_2_RX_RESET_N	epcs_rx_intf_2:rstn
SERDES_IF_0:EPCS_2_TX_RESET_N	epcs_tx_intf_2:rstn
SERDES_IF_0:EPCS_2_RX_CLK	epcs_rx_intf_2:clk
SERDES_IF_0:EPCS_2_TX_CLK	epcs_tx_intf_2:clk
epcs_rx_intf_1:rxdin[19:0]	delay1:out_data[19:0]
epcs_rx_intf_2:rxdin[19:0]	delay2:out_data[19:0]

Рис. 14. Диалоговое окно установки нового имени вывода для подключения к верхнему уровню

бывает полезно переименовать вывод верхнего уровня. Для этого следует щелкнуть по нему правой кнопкой мыши и выбрать из выпавшего меню пункт Rename Top Level Pin (рис. 14), а затем ввести в появившемся одноименном диалоговом окне новое имя.

Итак, выведем часть сигналов нашего модуля для подключения на верхнем уровне и переименуем некоторые внешние выводы (табл. 2). Предварительное замечание: если введено неправильное новое имя для вывода, то диалоговое окно Rename Top Level Pin после нажатия на кнопку OK закроется без вывода окна с сообщением об ошибке, но прежнее имя вывода не изменится. Сообщение об ошибке будет, но оно появится в поле протокола в нижней части окна Libero SoC, и его можно не заметить или вообще не увидеть, если поле протокола закрыто. Поэтому результат переименования надо контролировать. Подобный случай может произойти, например, если при копировании имени вывода через буфер обмена был случайно захвачен из исходного текста лишний пробел. И еще: при переименовании шины в окно Rename Top Level Pin следует вводить только имя шины, без квадратных скобок с номерами линий шины. Так, если новое имя шины busname [7:0], надо ввести только busname.

Следующие два входа модуля SERDES_IF_0 надо подключить к выводу верхнего уровня EPCS_RESET_N:

- EPCS_1_RESET_N;
- EPCS_2_RESET_N.

Для этого сначала создадим порт с именем EPCS_RESET_N. Выберем удобное пустое место на холсте, щелкнем по нему правой кнопкой мыши и выберем из выпавшего меню пункт Add Port. Появится диалоговое окно добавления порта (рис. 15).

Введем в поле Name наше имя порта, выберем его тип (в данном случае Input) и нажмем OK. После этого можно выбрать при нажатой

Рис. 15. Окно Add Port

Таблица 2. Список выводов для подключения на верхнем уровне

Модуль	Имя вывода	Новое имя
SERDES_IF_0	APB_S_PRESET_N	
SERDES_IF_0	APB_S_PCLK	
SERDES_IF_0	APB_SLAVE	
SERDES_IF_0	REFCLK1_OUT	
SERDES_IF_0	EPCS_1_READY	Lane1_READY
SERDES_IF_0	EPCS_1_RX_IDLE	Lane1_RX_IDLE
SERDES_IF_0	EPCS_1_RX_RESET_N	Lane1_RX_RESET_N
SERDES_IF_0	EPCS_1_TX_RESET_N	Lane1_TX_RESET_N
SERDES_IF_0	EPCS_1_RX_CLK	Lane1_RX_CLK
SERDES_IF_0	EPCS_1_TX_CLK	Lane1_TX_CLK
SERDES_IF_0	EPCS_2_RX_RESET_N	Lane2_RX_RESET_N
SERDES_IF_0	EPCS_2_TX_RESET_N	Lane2_TX_RESET_N
SERDES_IF_0	EPCS_2_RX_CLK	Lane2_RX_CLK
SERDES_IF_0	EPCS_2_TX_CLK	Lane2_TX_CLK
epcs_tx_intf_1	txdin[19:0]	Lane1_TX_data[19:0]
epcs_tx_intf_2	txdin[19:0]	Lane2_TX_data[19:0]
epcs_rx_intf_1	rxdout[19:0]	Lane1_RX_data[19:0]
epcs_rx_intf_2	rxdout[19:0]	Lane2_RX_data[19:0]
epcs_rx_intf_1	rxvalo	Lane1_RX_VAL
epcs_rx_intf_2	rxvalo	Lane2_RX_VAL
SERDES_IF_0	REFCLK1_OUT	REFCLK_OUT

клавише Ctrl новый порт и оба указанных вывода модуля SERDES_IF_0, щелкнуть по одному из них правой кнопкой мыши и выбрать из выпавшего меню пункт Connect.

Теперь надо пометить как неиспользуемые те из выводов модулей проекта, которые никуда не будут подключены. У нас это следующие выводы модуля SERDES_IF_0:

- EPCS_1_TX_CLK_STABLE;
- EPCS_2_READY;
- EPCS_2_RX_IDLE;
- EPCS_2_TX_CLK_STABLE,

а также выводы txvalo модулей epcs_tx_intf_1 и epcs_tx_intf_2.

Для этого можно выбрать их все при нажатой клавише Ctrl, щелкнуть по любому из них правой кнопкой мыши и выбрать из выпавшего меню пункт Mark Unused. В результате каждый из выводов будет отмечен косым крестиком.

Аналогично подадим низкий логический уровень на следующие выводы модуля SERDES_IF_0:

- EPCS_1_PWRDN;
- EPCS_1_TX_OOB;
- EPCS_1_RX_ERR;
- EPCS_2_PWRDN;
- EPCS_2_TX_OOB;
- EPCS_2_RX_ERR,

а также на выводы txvali модулей epcs_tx_intf_1 и epcs_tx_intf_2.

Для этого из выпавшего меню надо будет выбрать пункт Tie Low.

И наконец, подадим высокий логический уровень на выводы EPCS_1_TX_VAL и EPCS_2_TX_VAL. Для подключения к высокому логическому уровню в упомянутом выпадающем меню служит пункт Tie High.

Все, на этом создание модуля SmartDesign закончено. Теперь надо выполнить генерацию компонента из этого модуля. Выберем из главного меню среды Libero SoC пункт SmartDesign -> Generate Component. Если схема составлена без ошибок, то в поле протокола (Log) в нижней части окна Libero SoC появится информационное сообщение «'EPCS_SERDES' was successfully generated». В противном случае Libero SoC покажет окно с диагностическим сообщением и предложит посмотреть отчет об ошибках, который поможет их устранить.

Шаг 5. Завершение работы со SmartDesign

Перейдем от закладки EPCS_SERDES на закладку EPCS_Demo_Top. Выберем в правой верхней части окна Libero SoC закладку Design Hierarchy, найдем на ней только что созданный нами модуль EPCS_SERDES и перетащим его на холст EPCS_Demo_Top.

Затем добавим на холст пользовательские модули, написанные на языке описания аппаратуры. Найдем на закладке Design Hierarchy и перетащим на холст следующие модули:

- count_gen;
- count_check;
- Output_select;
- prbs7_16_check;
- prbs7_16_gen;
- FabUART.

При необходимости имена модулей можно изменять.

Теперь создадим необходимые соединения между компонентами на холсте (табл. 3).

Выведем для подключения на верхнем уровне следующие сигналы:

- COREUART_0: RX;
- FabUART_0: switch;
- FabUART_0: start;
- FabUART_0: connect_o;
- COREUART_0: TX.

Отметим, что это можно сделать и при помощи инструмента Quick Connect (его можно вызвать и нажатием сочетания клавиш Ctrl+K). В его выпадающем меню есть пункт Promote to Top Level.

Подадим низкий логический уровень на входы модуля CorePCS_0:

- FORCE_DISP [1:0];
- DSP_SEL [1:0].

Подадим низкий логический уровень на входы модуля COREUART_0:

- CSN;
- ODD_N_EVEN;
- PARITY_EN.

Подадим высокий логический уровень на вход WA_RSTn модуля CorePCS_0, а также на входы модуля EPCS_Demo_0:

- SDIF0_PERST_N;
- SDIF0_SPLL_LOCK,

а также на вход BIT8 модуля COREUART_0.

Отметим как неиспользуемые следующие выводы модуля COREUART_0:

- OVERFLOW;
- PARITY_ERR;
- FRAMING_ERR.

Таблица 3. Соединения между компонентами на холсте EPCS_Demo_top

От	До
CorePCS_0:RX_DATA[15:0]	count_check_0:data_in[15:0]
CorePCS_0:EPCS_TX_DATA[19:0]	EPCS_SERDES_0:Lane1_TX_data[19:0]
CorePCS_0:RX_K_CHAR[1:0]	count_check_0:k_in[1:0]
CorePCS_0:ALIGNED	count_check_0:rx_val
CorePCS_0:RESET_N:EPCS_TxRSTn	count_gen_0:reset_n
	EPCS_SERDES_0:Lane1_TX_RESET_N
CorePCS_0:EPCS_READY	EPCS_SERDES_0:Lane1_READY
CorePCS_0:EPCS_TxCLK	EPCS_SERDES_0:Lane1_TX_CLK
	count_gen_0:clk
CorePCS_0:EPCS_RxRSTn	EPCS_SERDES_0:Lane1_RX_RESET_N
	AND2_1:A
CorePCS_0:EPCS_RxCLK	EPCS_SERDES_0:Lane1_RX_CLK
	count_check_0:clk
CorePCS_0:EPCS_RxVAL	EPCS_SERDES_0:Lane1_RX_VAL
	Output_select_0:Rx_Val_L1
CorePCS_0:EPCS_RxIDLE	EPCS_SERDES_0:Lane1_RX_IDLE
CorePCS_0:EPCS_RxDATA[19:0]	EPCS_SERDES_0:Lane1_RX_data[19:0]
CorePCS_0:TX_DATA[15:0]	count_gen_0:data_out[15:0]
CorePCS_0:TX_K_CHAR[1:0]	count_gen_0:k_out[1:0]
EPCS_SERDES_0:Lane2_RX_RESET_N	AND2_0:A
count_gen_0:error_inject	prbs7_16_gen_0:error_inject
	FabUART_0:genrate_err
prbs7_16_gen_0:reset_n	EPCS_SERDES_0:Lane2_TX_RESET_N
prbs7_16_gen_0:clk	EPCS_SERDES_0:Lane2_TX_CLK
prbs7_16_gen_0:Data_out[19:0]	EPCS_SERDES_0:Lane2_TX_data[19:0]
EPCS_Demo_0:INIT_APB_S_PCLK	EPCS_SERDES_0:APB_S_PCLK
EPCS_Demo_0:INIT_APB_S_PRESET_N	EPCS_SERDES_0:APB_S_PRESET_N
EPCS_Demo_0:INIT_DONE	EPCS_SERDES_0:EPCS_RESET_N
COREUART_0:WEN	FabUART_0:uart_wen
COREUART_0:OEN	FabUART_0:uart_oen
COREUART_0:DATA_IN[7:0]	FabUART_0:uart_data_out[7:0]
COREUART_0:RXRDY	FabUART_0:uart_rxrdy
COREUART_0:TXRDY	FabUART_0:uart_txrdy
COREUART_0:DATA_OUT[7:0]	FabUART_0:uart_data_in[7:0]
FabUART_0:rx_val	Output_select_0:Rx_val_out
FabUART_0:rx_lock	Output_select_0:Lock
FabUART_0:rx_error	Output_select_0:Rx_error
FabUART_0:error_count[5:0]	Output_select_0:Error_out[5:0]
FabUART_0:start	Output_select_0:reset_n
	AND2_0:B
	AND2_1:B
FabUART_0:switch	Output_select_0:switch
FabUART_0:clear	count_check_0:clr_err_counter
	prbs7_16_check_0:clr_err_counter
AND2_0:Y	prbs7_16_check_0:reset_n
AND2_1:Y	count_check_0:reset_n
prbs7_16_check_0:clk	EPCS_SERDES_0:Lane2_RX_CLK
Prbs7_16_check_0:data_in[19:0]	EPCS_SERDES_0:Lane2_RX_data[19:0]
count_check_0:error_out	Output_select_0:RX_Error_L1
count_check_0:lock	Output_select_0:Lock_L1
count_check_0:error_count[5:0]	Output_select_0:Error_In_L1[5:0]
prbs7_16_check_0:error_out	Output_select_0:RX_Error_L2
prbs7_16_check_0:lock	Output_select_0:Lock_L2
prbs7_16_check_0:error_count[5:0]	Output_select_0:Error_In_L2[5:0]

Вспомним, что подать низкий логический уровень (Tie Low), высокий логический уровень (Tie High), а также отметить вывод как неиспользуемый (Mark Unused) можно и с помощью инструмента Quick Connect.

Для устройства M2GL010 следует отметить как неиспользуемые следующие порты модуля EPCS_Demo_0:

- SDIF0_PHY_RESET_N;
- SDIF0_CORE_RESET_N;
- POWER_ON_RESET_N;
- HPMS_READY;
- SDIF_READY;
- FAB_CCC_GL3;
- COMM_BLK_INT;
- HPMS_INTM2F [15:0].

Если же мы работаем с устройством M2S090, как неиспользуемые следует отметить порты:

- POWER_ON_RESET_N;
- SDIF_READY;
- MSS_READY;
- SDIF0_PHY_REST_N;
- SDIF0_0_CORE_RESET_N;
- SDIF0_1_CORE_RESET_N;
- FAB_CCC_GL1.

И наконец, отметим как неиспользуемые следующие порты модуля CorePCS_0:

- EPCS_PWRDN;
- EPCS_TXOOB;
- EPCS_TxVAL;
- EPCS_RxERR;
- INVALID_K [1:0];
- CODE_ERR_N [1:0];
- B_CERR [1:0];
- RD_ERR [1:0].

Теперь нам осталось выполнить генерацию компонента для модуля EPCS_Demo_top. Делается это точно так же, как для модуля EPCS_SERDES.

Шаг 6. Подключение файлов ограничений

Самостоятельное создание файлов ограничений в рамках лабораторной работы, очевидно, отнимет чересчур много времени и не слишком поможет достичь нашей цели — изучения основ работы с модулями SERDES. Поэтому корпорация Microsemi предлагает импортировать эти файлы из готового проекта, который можно найти на ее сайте в виде архива в формате ZIP [1]. Внутри архива лежат несколько папок с файлами готового проекта. Этот проект можно открыть, например, чтобы сверить с ним соединения, сделанные в предыдущих разделах. Файлы ограничений лежат в папке «<имя папки с распакованным архивом>\Libero Project\constraint\».

Перейдем на закладку Design Flow, найдем там раздел Create Constraints и развернем его. Щелкнем правой кнопкой мыши по строке I/O Constraints и выберем из выпадающего меню пункт Import Files. Через открывшееся диалоговое окно найдем нужную папку с файлами ограничений, в ней найдем папку io, а в ней — файл EPCS_IGL2_EB.pdc. Это так называемый файл физических ограничений. В нем описаны назначения для выводов ПЛИС. Затем щелкнем правой кнопкой мыши по строке Timing Constraints и аналогично импортируем в проект файл EPCS_Demo_top_compile.sdc, который лежит в папке с файлами ограничений. В нем находятся ограничения для компиляции и трассировки.

Шаг 7. Генерация данных для программирования

Теперь нам осталось сформировать данные для программирования, которые впоследствии можно будет загрузить в конфигурационную память ПЛИС. Найдем на закладке Design Flow строку Generate Bitstream и дважды щелкнем по ней левой кнопкой мыши. После этого среда Libero SoC произведет трассировку, проверку выполнения временных ограничений и сгенерирует данные для программирования.

Затем найдем на закладке Design Flow раздел Handoff Design for Production, в нем — строку Export Bitstream и дважды щелкнем по ней мышью. Так мы получим файл с расширением STP, который можно использовать для загрузки конфигурации при помощи программатора FlashPro4.

Вспомним первую часть этой статьи [1]. Там мы проверяли готовность нашего оборудования и программного обеспечения к выполнению лабораторной работы по изучению блоков SERDES. Статья заканчивалась на том, что мы брали готовый файл STP из распакованного архива [2], загружали его в ПЛИС на плате отладочного набора и смотрели, как эта плата общается с демонстрационной программой, работающей на персональном компьютере.

Загрузим только что полученный нами файл с расширением STP в ПЛИС и проверим: работает ли наш проект? Если нет, то необходимо вернуться и еще раз все проверить. Можно обратиться и к такому средству отладки, как SmartDebug (кнопка Debug SERDES).

Шаг 8. Создание клиента ENVM для SmartFusion2

При работе с устройствами SmartFusion2 нам необходимо создать встроенное программное обеспечение для микроконтроллера ARM Cortex-M3, которое будет инициализировать элементы микроконтроллерной подсистемы (Microcontroller Subsystem, MSS) и интерфейс SERDES при помощи IP-ядра CoreConfigP. Процессор Cortex-M3 выполняет код eNVM после сброса устройства SmartFusion2.

Это достаточно объемная работа, а потому в рамках данной статьи мы ее рассматривать не будем. Ее описание можно найти в тексте [2], на основе которого написана данная статья.

Приложение. Применение учебного проекта для пользовательских разработок

Структура нашего учебного проекта предусматривает возможность создания на его основе пользовательских проектов с требуемой функциональностью (рис. 16).

Генератор псевдослучайной двоичной последовательности (ПСДП) PRBS7 в секции передатчика можно заменить пользовательским генератором данных. Он будет формировать сигналы для интерфейсного блока передатчика. Вместо схемы проверки ПСДП в приемнике можно использовать пользовательский приемник, который будет получать данные из интерфейсного блока приемника.

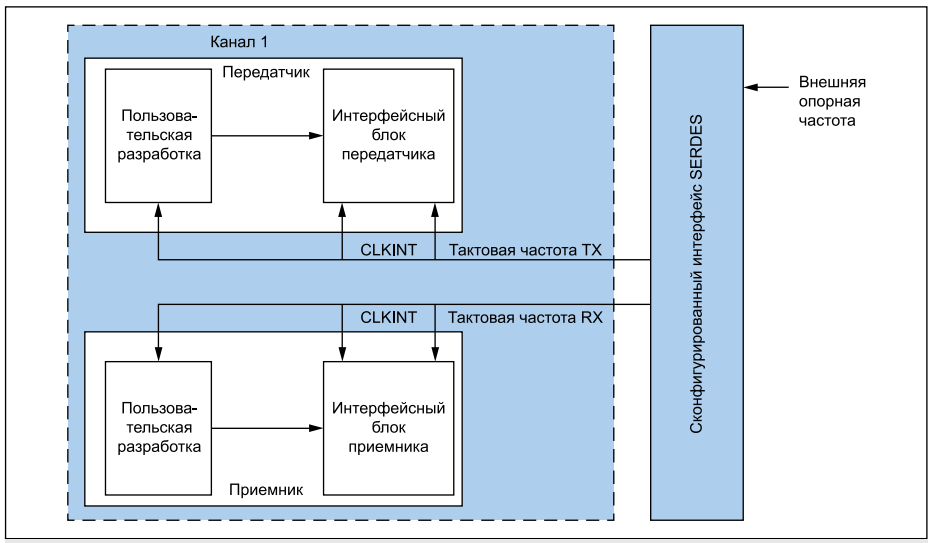

Рис. 16. Блок-схема учебного проекта

Заключение

В двух последних статьях цикла, посвященного новым семействам ПЛИС корпорации Microsemi, мы рассмотрели применение высокоскоростных блоков SERDES, которые встроены в них. По своим параметрам эти блоки являются одними из лучших в отрасли. Интегрированная САПР Libero SoC предоставляет удобные и интуитивно понятные средства для работы с SERDES.

Более подробную информацию по теме можно найти в [3], в документации на IP-ядра, использованные в учебном проекте, и других материалах на сайте корпорации Microsemi в разделе, посвященном ПЛИС и системам на кристалле [3].

Литература

1. Иоффе Д., Казаков А. Блоки SERDES в новых ПЛИС корпорации Microsemi. Часть 1 // Компоненты и технологии. 2015. № 1.
2. **http://soc.microsemi.com/download/rsc/?f=IGLOO2_EPCS_Demo_11p4_sp1_DF** — архив с файлами учебного проекта.
3. Implementing a SmartFusion2/IGLOO2 SERDES EPCS Protocol Design — Libero SoC v11.4 Tutorial.
4. **http://www.microsemi.com/products/fpga-soc/fpga-and-soc**