Продолжение. Начало в КиТ № 7'2005

Обзор маршрута проектирования ПЛИС FPGA Advantage компании Mentor Graphics. Часть 6. Подготовка к тестированию

Алексей РАБОВОЛЮК
al1@megratec.ru

В продолжении описания маршрута проектирования FPGA Advantage, начатого в № 7`2005, речь пойдет о создании тестбенча для отлаженного и синтезированного регистра BCDRegister. Тестбенч будет выполнен в виде блок-схемы (Flow Chart).

Для полноценной верификации проекта, как правило, применяется так называемый тестбенч. Тестбенч — это некий виртуальный объект, содержащий как исходный тестируемый проект, так и блок генерации тестовых воздействий (рис. 1). Виртуальность тестбенча заключается в том, что он предназначен исключительно для моделирования и, соответственно, может включать несинтезируемые конструкции VHDL и Verilog. Перед созданием тестбенча необходимо выполнить несколько подготовительных действий:

1. Создать библиотеку, в которой будут храниться элементы тестбенча.
2. Если исходный проект представлен в виде блока, то конвертировать его в компонент (см. «КиТ» № 8`2005).
3. При необходимости отредактировать символ верхнего уровня.

Блок генерации тестовых воздействий (далее просто «тестер») имеет обратную связь с исходным проектом, следовательно, генерируемые входные воздействия могут зависеть от результатов работы проекта.

Как правило, тестер реализуется в виде блока, так как блок имеет гибкий интерфейс и может быть создан «сверху», то есть добавлен в виде нового пустого блока на блок-диаграмму (Block Diagram) (см. «КиТ» № 8`2005). Кроме того, может возникнуть необходимость переделать тестер «на лету», а архитектура может быть в виде любого доступного в среде HDL Designer представления архитектуры. Наиболее простым вариантом реализации архитектуры является блок-схема (Flow Chart). При необходимости блок может быть преобразован в компонент. Для этого необходимо щелкнуть правой кнопкой мыши на блоке и в появившемся меню выбрать Convert Block to Component.

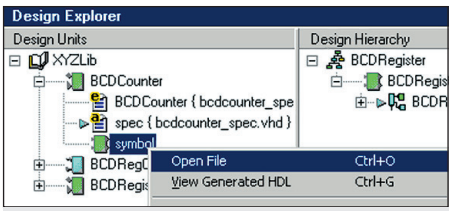

Рис. 2. Открытие символа компонента на редактирование из окна Design Explorer

Если есть необходимость отредактировать символ исходного проекта, то это можно сделать в редакторе символа. Редактор открывается несколькими способами:

- из окна Design Explorer: щелкнуть правой кнопкой мыши на нужном символе и выбрать пункт Open File (рис. 2);
- из окна редактора блок-диаграмм (Block Diagram): щелкнуть правой кнопкой мыши на компоненте и выбрать Open As > Symbol (рис. 3).

Редактор символа может работать в одном из двух режимов: графическом и табличном (рис. 4). Графический режим интуитивно более понятен, а табличный полнее отображает параметры символа и более пригоден для документирования. Переключение между режимами осуществляется при помощи панели Structure Navigator.

В графическом режиме непосредственно на символе, как правило, отображается следующая информация: имя библиотеки, в которой хранится символ, имя объекта, которому при-

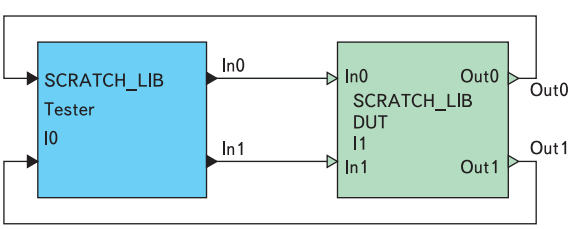

Рис. 1. Пример архитектуры тестбенча

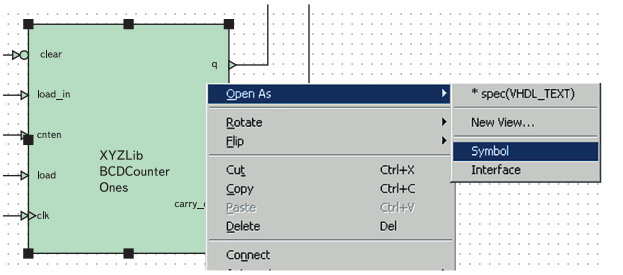

Рис. 3. Открытие символа компонента на редактирование из редактора блок-диаграммы

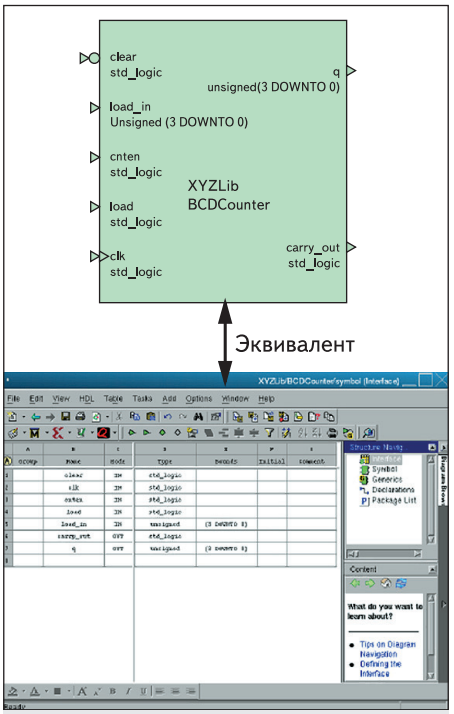

Рис. 4. Режимы работы редактора символа: графический и табличный

надлежит символ, и данные о портах. Какая именно информация будет отображаться, а также ее формат можно указать в окне Symbol Properties (которое можно открыть, щелкнув в редакторе символа правой кнопкой мыши на изображении символа и выбрав Object Properties). Если к порту необходимо добавить символ инверсии, символ синхросигнала или изменить режим работы порта (вход, выход), то необходимо щелкнуть правой кнопкой мыши по изображению порта и выбрать соответствующий пункт меню. Форма символа может быть отредактирована произвольным образом, вплоть до включения растрового графического файла. Для этого в редакторе символа имеются все необходимые инструменты редактирования графики. Панели с соответствующими кнопками включаются через меню View > Toolbars > Apperance, View > Toolbars > Arrange Object и View > Toolbars > Comment Graphics. Помимо ручного редактирования можно воспользоваться набором готовых форм, доступных через меню Diagram > AutoShapes. При наличии большого количества портов можно воспользоваться функцией автоматического выравнивания расстояний между портами на каждой из сторон символа. Функция доступна через меню Diagram > Equidistant Ports. По умолчанию граница символа совпадает с его формой. Разделить их можно при помощи меню Diagram > Custom Symbol. При этом граница символа будет показана пунктирной линией, что позволит перемещать ее независимо от формы, а форму можно будет составлять из нескольких графических элементов.

Сохранить отредактированный символ можно как в рамках существующего объекта проектирования File > Save, так и инициировать создание нового объекта проектирования File > Save As. Во втором случае будет необходимо указать имя существующей библиотеки, в которой будет создан новый объект проектирования, его имя и имя нового символа.

После сохранения символа все вновь добавляемые на блок-диаграмму вхождения отредактированного объекта проектирования будут отображаться с новым символом, а ранее созданные вхождения будут отображаться со старым символом. Для того чтобы изменения в символе распространились на все вхождения объекта проектирования, содержащего этот символ, необходимо в редакторе символа щелкнуть правой кнопкой мыши по изображению символа и в появившемся меню выбрать пункт Update Where Used. В появившемся диалоговом окне необходимо указать место, где нужно производить поиск, и нажать кнопку Start. Будет произведен поиск вхождений отредактированного объекта проектирования и подстановка нового символа. В случае, если при редактировании символа произошли изменения в портах (список, тип и т. д.), то процедуру обновления вхождений необходимо выполнить в обязательном порядке. Формы символа объекта проектирования и символа вхождения этого объекта могут различаться.

Заготовку тестбенча можно сгенерировать автоматически. Для этого необходимо во вложенном в Design Manager окне Design Explorer выделить объект проектирования, для которого будет создан тестбенч, и выполнить команду File > New > Test Bench. При этом будет создан новый объект проектирования, представлением которого будет блок-диаграмма, и эта блок-диаграмма будет содержать исходный объект проектирования (в виде компонента) и подключенный к нему пустой блок — тестер. Далее необходимо добавить новую архитектуру к пустому блоку тестера, щелкнув на этом блоке правой кнопкой мыши и выбрав Open As > New View.

Самым простым вариантом реализации тестера является блок-схема. Блок-схема (рис. 5) представляет собой последовательность действий (actions) плюс элементы ветвления (conditions). Принцип построения схемы соответствует принципам написания конструкции Process в языке VHDL. Например, в соответствии с языком VHDL, процесс может быть описан одним из двух стилей: либо с указанием списка чувствительности, либо с использованием предложений wait внутри процесса (подробности — в любом учебнике по VHDL). Задать список чувствительности можно на вкладке Generation окна Flow Chart Properties, которое можно открыть при помощи меню Diagram > Flow Chart Properties в окне редактора блок-схемы. В этом же окне Flow Chart Properties можно объявить и определить

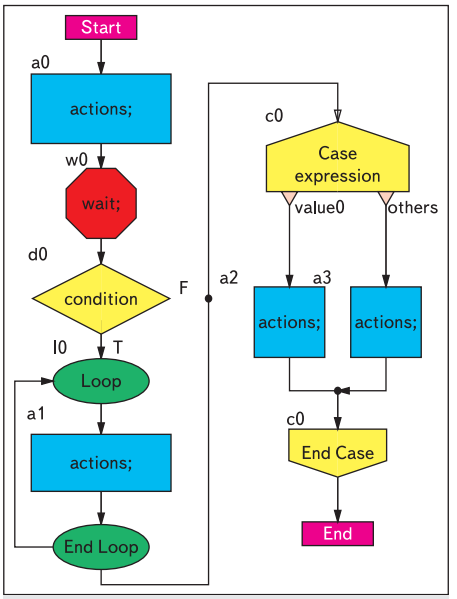

Рис. 5. Пример блок-схемы

внутренние переменные для данной блок-схемы (вкладка Architecture Declarations), параллельные предложения (вкладка Concurrent Statment) и внутренние переменные для процесса (вкладка Process Declarations). Кроме того, на вкладке Generation можно задать различные настройки для генерации HDL.

На блок-схеме могут присутствовать следующие элементы (рис. 6):

- Действия (Action Box).
- Условное ветвление (Decision Box).
- Предложения wait (Wait Block).
- Точки начала и окончания (Start/End Point).
- Начало и окончание цикла (Loop Start/End).
- Множественное ветвление (Case Box).
- Соединения (Flow Arrow).

Эти элементы могут быть добавлены на схемы как при помощи меню Add > <имя_элемента>, так и при помощи соответ-

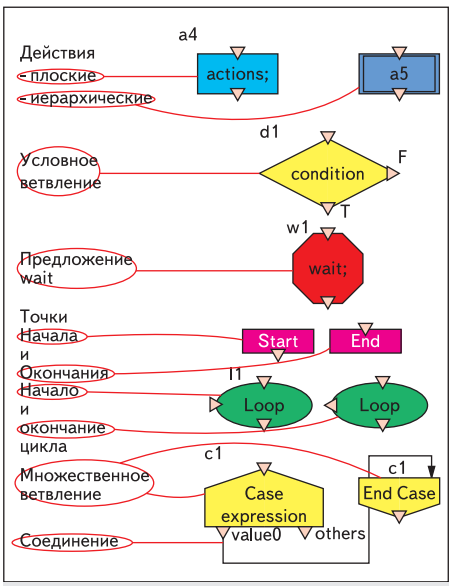

Рис. 6. Элементы блок-схемы

ствующей панели инструментов, расположенной непосредственно над схемой.

Действия представляют собой предложения на языке VHDL или Verilog, которые необходимо прописывать с полным соблюдением синтаксиса. Например, все VHDL-предложения должны заканчиваться точкой с запятой. Внутри иерархического действия может находиться вложенная блок-схема.

Внутрь условного ветвления необходимо вписать HDL-предложение, представляющее собой логическую функцию. При использовании языка VHDL содержимое желтого ромба будет вставлено в конструкцию IF, следовательно, ставить точку с запятой в конце логической функции не следует.

Предложения wait останавливают выполнение процесса. В языке VHDL предложение wait может быть описано в соответствии с одним из нескольких стилей. Например, команда «wait ;» (без кавычек) остановит процесс навсегда, «wait on 25 ns;» остановит выполнение процесса на 25 наносекунд. Для разных стилей уже существуют заготовки. Чтобы их увидеть, необходимо, щелкнуть правой кнопкой мыши на красном ромбе и в появившемся меню выбрать Object Properties. В нижней части появившегося окна будут перечислены заготовки для различных стилей предложения wait.

Только одна точка начала и как минимум одна точка окончания процесса должна присутствовать на блок-схеме. При использовании стиля «со списком чувствительности» при пересчете в процессе моделирования хотя бы одной переменной из «списка» схема будет выполняться от точки начала до точки окончания. При использовании стиля «с предложениями wait» при достижении точки окончания работа схемы сразу продолжится с точки начала, а прерывание работы будет выполняться при достижении предложения wait.

Циклы работают так же, как и в других языках программирования: в точке начала цикла задается условие, при нарушении которого будет выполнен выход из цикла.

При использовании языка VHDL на множественное ветвление накладывается ограничение: все возможные варианты значений, которые может принять условное выражение (case expression), должны быть перечислены в «выходах». Если перечислены не все варианты, то обязательно должен присутствовать выход others. Более подробно о работе конструкции case можно прочитать в любом учебнике по VHDL.

Соединение представляет собой линию со стрелкой, показывающую очередность выполнения элементов схемы.

Объект проектирования, представлением которого является блок-схема, на самом деле может содержать внутри себя несколько «параллельных» (то есть одновременно выполняющихся) блок-схем. Эти параллельные блок-схемы называются процессами. Список

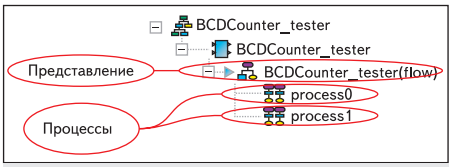

Рис. 7. Процессы в блок-схеме

всех параллельных процессов можно увидеть во вложенном окне Strucrure Navigator, находящимся в правом верхнем углу редактора блок-схем (рис. 7). Добавить новый процесс можно, щелкнув правой кнопкой мыши в строке Concurrent FlowChart и в появившемся меню выбрав New. Удалить или переименовать процесс можно аналогичным образом, щелкнув на имени интересующего процесса и выбрав соответствующую команду.

Самостоятельная работа

Если в распоряжении пользователя имеются сконфигурированные данные для лабораторных работ, о которых говорилось в первой части статьи (см. «КиТ» № 7`2005), создана блок-диаграмма (см. «КиТ» № 8`2005), создана архитектура для блока BCDRegControl (см. «КиТ» № 9`2005), объект BCDRegister без ошибок промоделирован в ModelSim (см. «КиТ» № 4`2006) и успешно синтезирован (см. «КиТ» № 6`2006), то можно приступить к созданию тестбенча для объекта BCDRegister.

Для того чтобы BCDRegister было удобно использовать в тестбенче, первое, что необходимо сделать, — это отредактировать внешний вид символа BCDRegister-а.

Одним из важных параметров читабельности символа является список отображаемой на нем информации. По умолчанию на символе отображается имя библиотеки, в которой он содержится, имя объекта, которому символ принадлежит, и имена портов. Информативность символа повысится, если кроме имен портов будут показаны и их типы. Включить отображение типов можно в редакторе символа, щелкнув правой кнопкой мыши по сим-

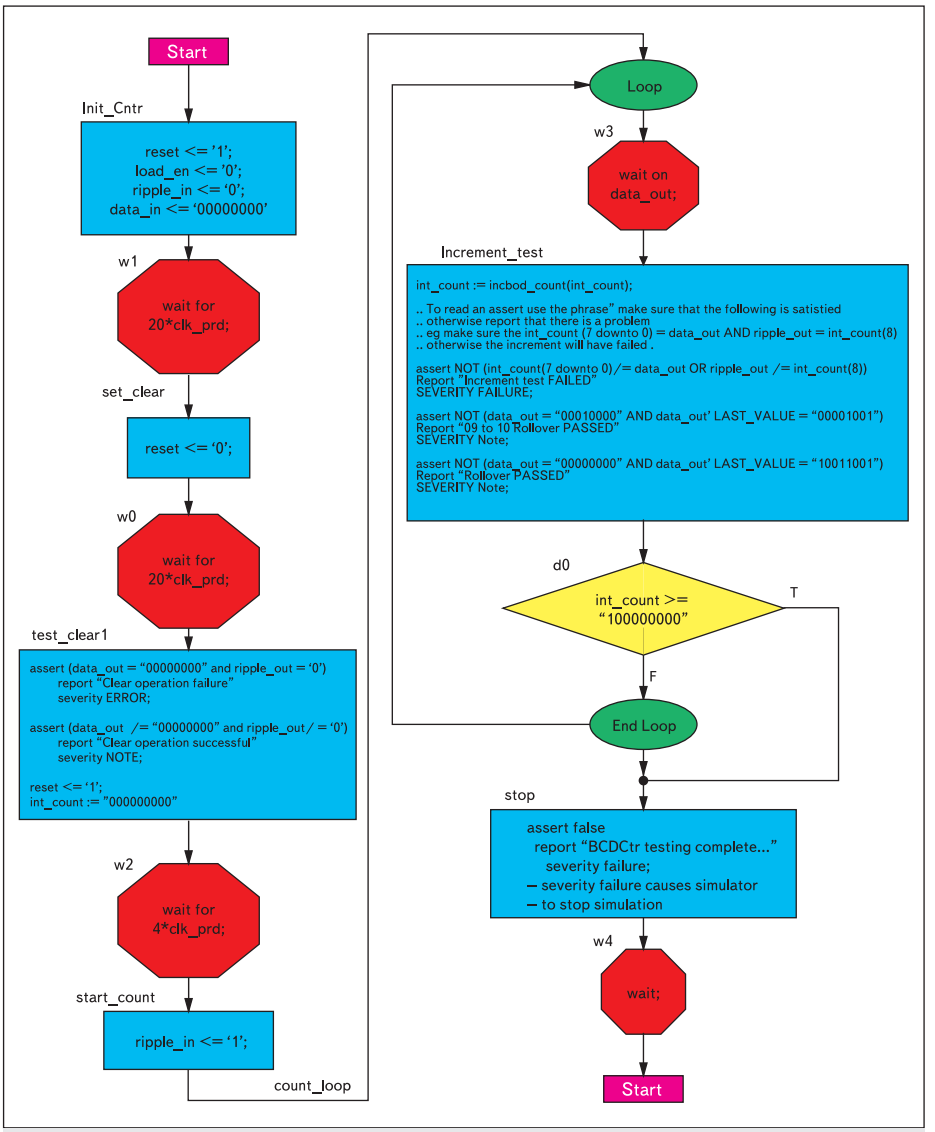

Рис. 8. Процесс Tester для самостоятельной работы

волу и в открывшемся меню выбрав Object Properties, затем в появившемся окне на вкладке Symbol нажать кнопку Port Display и в новом окне задать формат отображения портов. Для того чтобы изменить настройки по умолчанию для портов, то есть настройки для всех символов, а не для одного выбранного, необходимо в окне Design Manager выбрать меню Options > Master Preferences > Interface, в левом дереве появившегося окна выбрать Symbol > Display Settings > VHDL Port Display и задать настройки по умолчанию для отображения портов на символе. Теперь вновь создаваемые символы будут отображаться в соответствии с новыми настройками. Для того чтобы привести уже существующий символ к виду по умолчанию, необходимо в редакторе символа выбрать меню Options > Master Preferences > Apply to New and Existing Objects.

Для создания тестбенча необходимо во вложенном окне Design Explorer окна Design Manager выделить объект BCDRegister и использовать меню File > New > Test Bench. В появившемся окне необходимо указать различные параметры создаваемого тестбенча, такие как имя тестбенча, имя блока генерации тестов (тестера), библиотеку в которой будет создан тестбенч, язык (VHDL или Verilog) и так далее, а затем нажать кнопку OK.

Например, можно использовать следующие значения полей:

- Test Bench Library — MyTestLib.
- Test Bench Design Unit — BCDRegister_tb.
- Name Of Tester block — BCDRegister_tester.
- Language — VHDL.
- Остальные пункты — оставить по умолчанию.

После вышеперечисленных операций в библиотеке MyTestLib появится объект проектирования BCDRegister_tb, который можно увидеть в окне Design Explorer и открыть на редактирование, дважды щелкнув по нему левой кнопкой мыши.

Для создания тестовых воздействий необходимо в редакторе блок-диаграммы BCDRegister_tb дважды щелкнуть левой кнопкой мыши по пустому блоку BCDRegister_tester. В появившемся окне необходимо выбрать тип блока (поле File Types) Flow Chart, а затем нажать кнопки Next и Finish.

Откроется редактор новой блок-схемы (Flow Chart). Переименуйте текущий процесс в Tester при помощи меню Diagram > Rename Flow Chart.

Опишите генератор тестовых воздействий (тестер) в соответствии с рис. 8. При наличии сконфигурированных данных для лабораторных работ, о которых говорилось в первой части статьи (см. «КиТ» №7`2005), для заполнения блока test_clear1 можно воспользоваться файлом test_clear1.txt из директории Lab_Templates, а для заполнения блока increment_test — файлом increment_test .txt из этой же директории.

Созданный только что тестер использует функцию конвертирования десятичного числа в двоично-десятичное. Эту функцию необходимо где-то описать. Это можно сделать, выбрав пункт меню Diagram > Flow Chart Properties, а в открывшемся окне — вкладку Process Declarations. В поле для ввода текста необходимо ввести содержимое файла function_incbcd_count.txt из директории Lab_Templates.

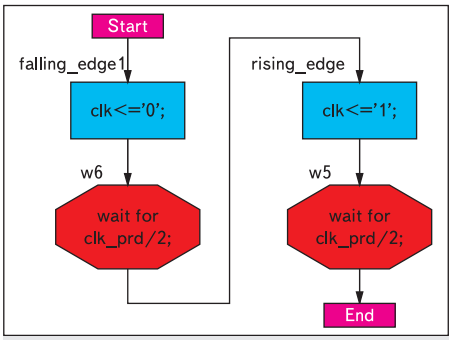

Рис. 9. Процесс — генератор синхросигнала для самостоятельной работы

Кроме непосредственно тестовых воздействий, тестбенч также должен генерировать синхросигнал. Для этого необходимо создать новый параллельный процесс. Это можно сделать при помощи меню Add > Concurrent Flow Chart. Реализуйте генератор синхросигнала частотой 20 МГц, как показано на рис. 9.

В генераторе синхросигнала используется переменная clk_prd. Чтобы ее инициализировать, воспользуйтесь меню Diagram > Flow Chart Properties и в закладке Architecture Declarations введите: «CONSTANT clk_prd: TIME := 50 ns;» (без кавычек).

Сохраните и закройте BCDRegister_tester. Перейдите в окно редактора тестбенча BCDRegister_tb и используйте меню Tasks > Generate > Run Through Components. Убедитесь, что генерация проходит без ошибок. Если это не так, постарайтесь исправить ошибки.

В следующей статье речь пойдет о расширенных методах отладки проекта в ModelSim.