Обзор маршрута проектирования ПЛИС FPGA Advantage компании Mentor Graphics.

Часть 8. Расширенные возможности проектирования

Алексей РАБОВОЛЮК
al1@megratec.ru

Данная статья является продолжением описания маршрута проектирования FPGA Advantage, начатого в КиТ № 7 `2005. В ней речь пойдет о расширенных возможностях проектирования, не рассмотренных в предыдущих статьях.

Параметр настройки — Generic. Язык VHDL позволяет описывать некий специфический элемент под названием generic. Условно его можно представить как переменную, подключенную к какому-либо элементу, например к entity (рис. 1). То есть это некий параметр элемента, имеющий имя и значение. В отличие от переменной, этот параметр настройки предназначен для влияния на процесс компиляции и синтеза. Например, при описании входного порта Q можно использовать следующую конструкцию:

```
q : IN unsigned (3 DOWNTO 0);
```

Однако если создать параметр настройки с именем razr и присвоить ему значение 3, то описанная выше конструкция примет следующий вид:

```
q : IN unsigned (razr DOWNTO 0);
```

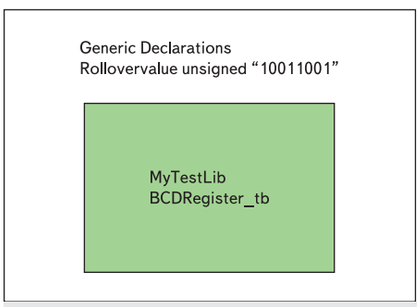

Рис. 1. Отображение параметра настройки на блок-диаграмме

Теперь при моделировании и синтезе можно легко менять «разрядность» проекта, просто меняя параметр настройки razr.

HDL Designer позволяет подключать параметры настройки к символам и блокам. Как уже было сказано выше, параметр настройки может быть подключен к entity. Именно этим обусловлен тот факт, что HDL Designer позволяет подключать параметры настройки только к символу (но не к вхождению символа на блок-диаграмме, то есть не к компоненту) и к блоку. В случае подключения параметра настройки к символу на вхождении этого символа в блок-диаграмму можно переопределить значение параметра настройки. Параметры настройки не являются иерархическими — созданный для конкретного символа, этот параметр настройки будет доступен только внутри архитектуры, подключенной к данному символу. Более подробно о теории параметров настройки можно узнать, обратившись к учебнику по VHDL.

Для того чтобы добавить параметр настройки к символу, необходимо открыть этот символ на редактирование и во вложенном окне Structure Navigator выбрать пункт Generics (рис. 2). При этом в основном поле редактора символа появится таблица параметров настройки. Если вложенного окна нет, то его можно включить в меню View > Diagram Browser. Столбец Value не обязателен для заполнения, так как значение может быть задано на вхождении этого символа в блок-диаграмму. Однако если при моделировании или синтезе значение так и не будет задано, то возникнет ошибка.

Для переопределения параметра настройки на вхождении символа в блок-диаграмму (то есть на компоненте) необходимо открыть редактор блок-диаграммы, щелкнуть правой кнопкой мыши по нужному компоненту и выбрать Object Properties. В появившемся окне редактора опций необходимо выбрать пункт Generics (рис. 3). В появившейся таблице можно изменять значения только ранее созданных параметров настройки.

Создавать параметры настройки в блоке можно в том же окне Object Properties. В данном случае таблица будет полностью доступна для редактирования.

Как уже упоминалось выше, параметры настройки не являются иерархическими. Рассмотрим пример (рис. 4). На верхнем уровне проекта, представленном компонентом BCDRegister_tb, создан параметр настройки xxx, и там же ему присвоено значение 100. Теперь, в пределах блок-диаграммы BCDRegister_tb будет доступен параметр настройки xxx. Если есть необходимость использовать тот же параметр настройки, с тем же значением внутри содержащегося на блок-диаграмме иерархического компонента BCD_Register, то необходимо не просто создать в символе элемента BCD_Register параметр настройки с таким же именем xxx, но и присвоить ему значение того параметра настройки, который «пришел» с верхнего уровня. Другими словами, в символе компонента BCD_Register необходимо создать параметр настройки с именем xxx (на рис. 4 ему присвоено значение 60), а на вхождении этого символа в блок-диаграмму присвоить этому параметру настройки значение, «пришедшее» с верхнего уровня, то есть xxx.

Возможность вставки в графику HDL-текста. Все графические архитектуры в HDL Designer позволяют включать HDL-текст. Это может понадобиться в том случае, если разработчику, например, необходимо объявить

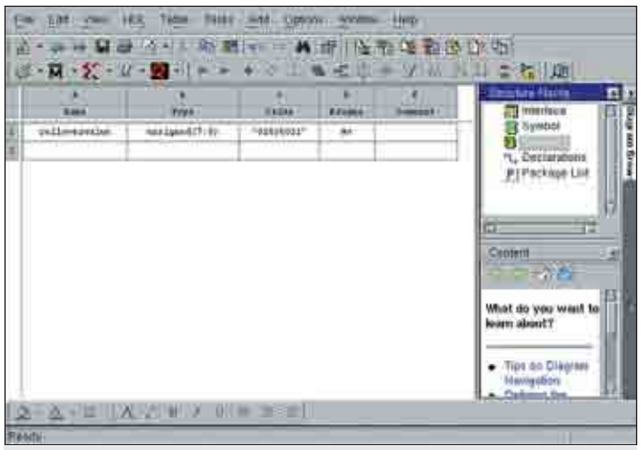

Рис. 2. Создание параметра настройки в редакторе символа

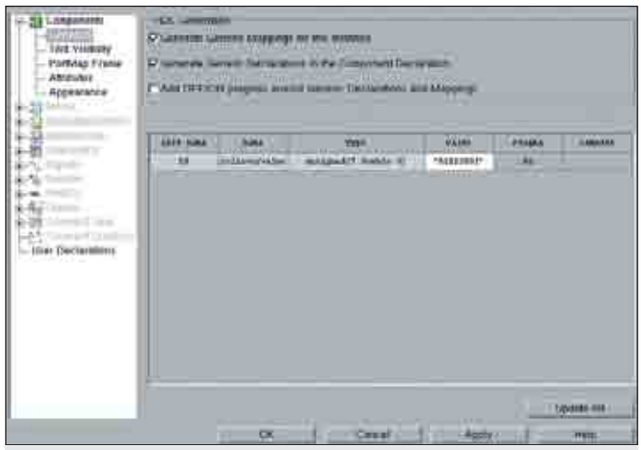

Рис. 3. Изменение значения параметра настройки в окне свойств компонента

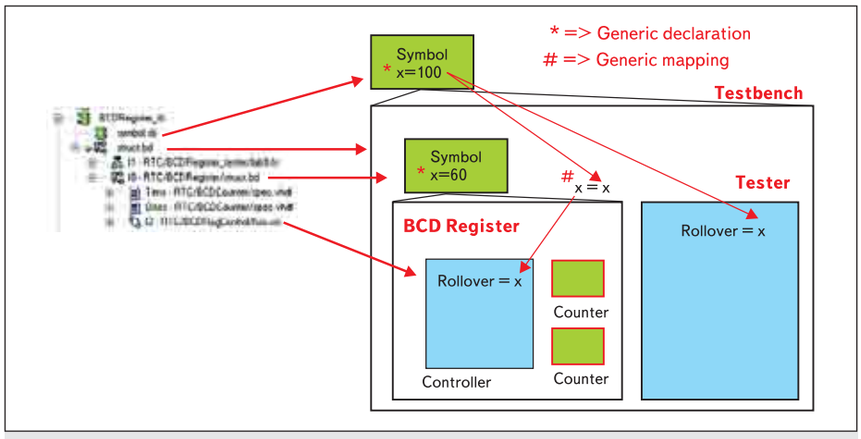

Рис. 4. Распространение параметра настройки через иерархию

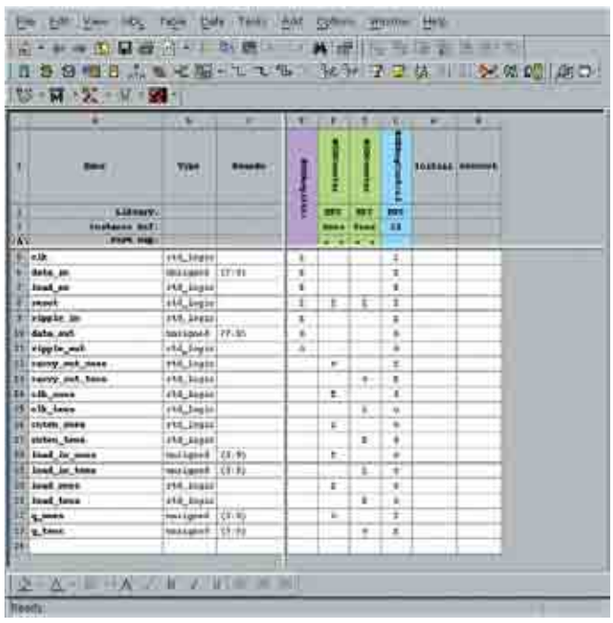

Рис. 5. Табличное представление блок-диаграммы

новые типы, создать новые внутренние сигналы или параллельные процессы, описать функции.

Блок-диаграмма позволяет вставлять текст двумя способами: через элемент «встроенный блок» (embeded block) (рассмотрен в одной из предыдущих статей) и через вкладку User Declarations во вложенном окне Structure Navigator.

Конечные автоматы (State Machines), блок-схемы (Flow Charts) и таблицы истинности (Truth Tables) содержат соответствующие вкладки в окнах свойств диаграмм. Так, например, в редакторе конечного автомата при выполнении команды меню Diagram > State Machine Properties появится окно настроек свойств конечного автомата, содержащее вкладки Statment Blocks и Declaration Blocks, выбрав которые, можно вставить HDL-текст в соответствующих полях.

Альтернативный способ представления блок-диаграммы — IBD (Interface Based Design). При выполнении в редакторе блок-диаграммы команды меню Diagram > Edit As IBD формат отображения диаграммы изменяется с графического на табличный (рис. 5). В табличном виде в левом столбце перечислены все сигналы, присутствующие на диаграмме, а в первой строке перечислены все вхождения элементов на блок-диаграмме. На пересечениях строк и столбцов обозначено, как сигнал взаимодействует с блоком: буквой I помечен входной сигнал, буквой O — выходной. Кроме перечисленных, IBD содержит и другие, интуитивно понятные данные. Этот способ представления блок-диаграммы тоже является полноценным редактором, то есть в нем можно производить те же действия, что и в графическом: создавать, удалять, редактировать сигналы, блоки, свойства. Таблица более удобна для документирования, а в некоторых случаях — и для редактирования.

Кроме IBD, HDL Designer позволяет создавать и обычные таблицы: таблицы истинности — Truth Tables. Для этого при создании новой архитектуры для компонента или бло-

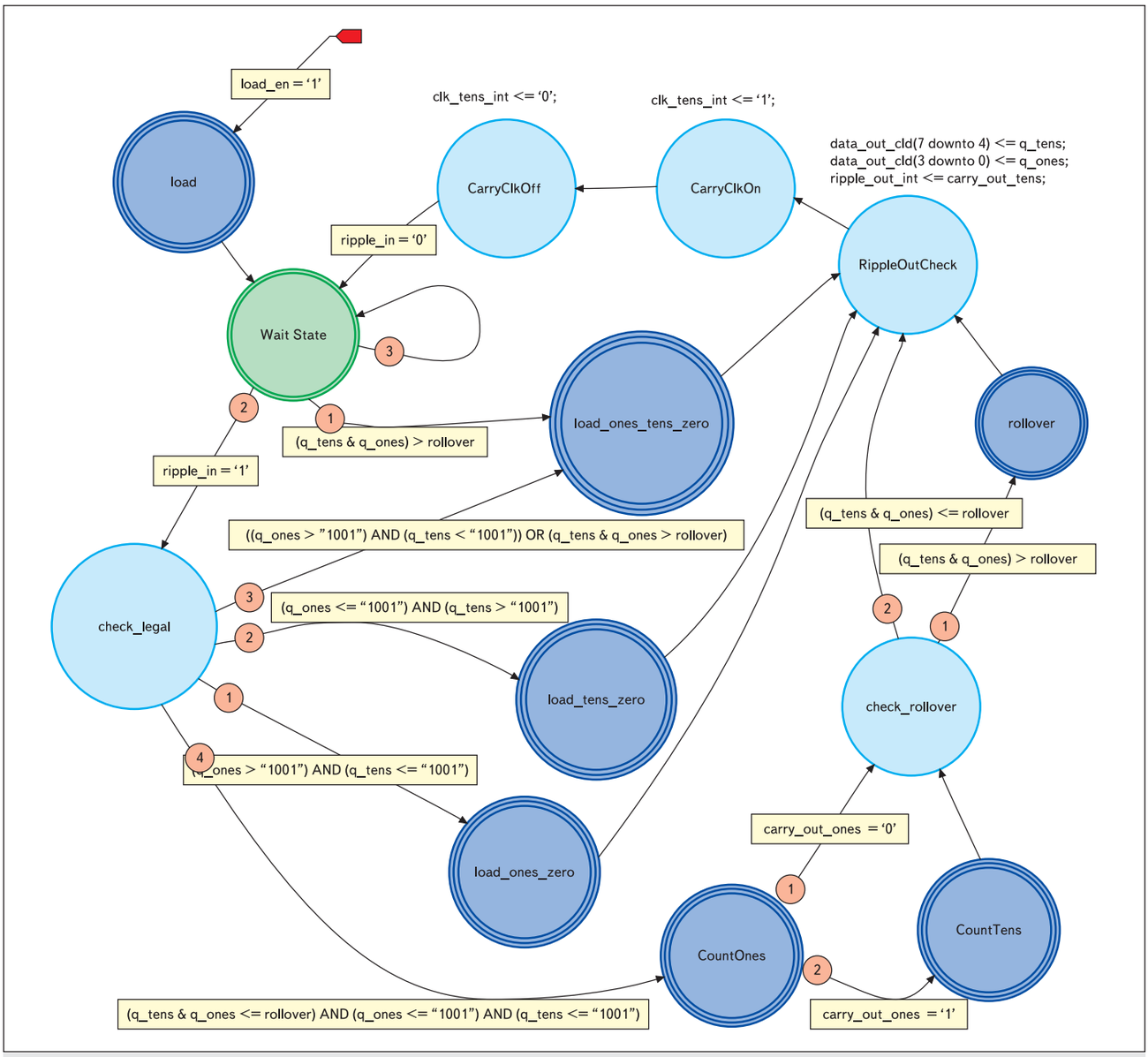

Рис. 6. Доработанный BCDRegControl

ка нужно указать тип Truth Table. Таблица истинности является интуитивно понятной: в левых столбцах перечислены входные порты, в правых — выходные, а в строках указывается, какие значения должны принимать выходные сигналы при указанных входных. Если входной сигнал в конкретной строке не имеет значения, то соответствующая клетка оставляется пустой. Если пустой остается вся строка входных сигналов, то это можно интерпретировать как «во всех остальных случаях». Возможности редактора таблиц истинности сравнимы с любым офисным редактором таблиц: копирование, перетаскивание, заполнение и т. д.

HDL Designer позволяет управлять элементами проекта так же, как обычными файлами в проводнике Windows. Для этого необходимо щелкнуть правой клавишей мыши по нужному объекту в окне Design Explorer и выбрать необходимое действие: cut, copy, paste, delete, rename и т. д. Очевидно, что, в отличие от файлов, работа со сложными объектами имеет свою специфику. Так, например, при переименовании компонента его вхождение в блок-диаграмму будет продолжать ссылаться на старое имя, и проект станет не пригодным для моделирования и синтеза. Для решения данной проблемы при попытке переименовать компонент HDL Designer предложит произвести поиск всех вхождений компонента в заданных библиотеках и коррекции ссылки на новое имя. При копировании компонента необходимо сначала скопировать его в буфер обмена (copy), открыть целевую библиотеку и, щелкнув правой кнопкой мыши на имени библиотеки, выбрать paste special, при этом появится диалоговое окно с дополнительными вопросами. В нем можно указать, копировать ли всю иерархию проекта (Copy Hierarchy), и если да, то какие именно иерархические элементы копировать (Components, Packages и т. д.). При отключенной кнопке Copy Hierarchy будет скопирован только символ и архитектура выбранного компонента, а все входящие в его состав иерархические элементы будут ссылаться на исходные библиотеки. В отличие от копирования компонента, при копировании блока (помечен синим) есть одна особенность: результатом копирования всегда будет компонент. Это связано со свойством блока, о котором говорилось в одной из предыдущих статей: блок может существовать только в единственном экземпляре. При выполнении

большинства действий элементы не должны быть открыты на редактирование. Так, при неудачной попытке переместить элемент (если выдано сообщение о том, что элемент заблокирован) необходимо убедиться, что данный элемент не открыт в окне редактора. Так как HDL Designer позволяет работать над проектом одновременно нескольким людям, то элемент может быть заблокирован любым из участников команды.

Если в распоряжении пользователя имеются сконфигурированные для лабораторных работ данные, о которых говорилось в первой статье, создана блок-диаграмма, описанная во второй статье, создана архитектура для блока BCDRegControl, описанная в третьей статье, объект BCDRegister без ошибок промоделирован в ModelSim, как это было описано в четвертой статье, успешно синтезирован, как это было описано в пятой статье, создан тестбенч, как это было описано в шестой статье, BCDRegister отлажен при помощи тестбенча, как это было описано в седьмой статье, то можно приступить к дальнейшей разработке проекта, которая будет заключаться в добавлении возможности BCDRegister сбрасываться в ноль после заданного в параметре настройки rollover числа и добавлении возможности загрузки в BCDRegister произвольного числа.

Первое, что необходимо будет сделать, — это добавить возможность загружать 8-разрядное число в BCDRegister. Данная функция берет данные с шины data_in и загружает их в регистры BCD Counters на переднем фронте синхросигнала при условии, что сигнал load_en равен 1. Старшие разряды записываются в counter Tens, младшие — в counter Ones. При загрузке необходимо проверять правильность загружаемых данных, и, если они не корректны, то сбрасывать BCDRegister в 0000.

Далее надо будет добавить к символу BCDRegister параметр настройки rollover. Этот параметр будет задавать число, после которого счетчик BCDRegister будет сбрасываться в 0000. Значение по умолчанию — 99, но в процессе моделирования надо будет поэкспериментировать с разными значениями.

Для экономии времени можно воспользоваться уже готовой архитектурой, реализующей механизм загрузки данных. Она находится в библиотеке Lab_Templates. Скопируйте архитектуру fsm_complete.sm из элемента BCDRegControl в ваш BCDRegControl и назначьте ее архитектурой по умолчанию. Это можно сделать, щелкнув на ней правой кнопкой мыши и выбрав пункт Set Default View. Откройте доработанный конечный автомат. Он должен выглядеть так, как показано на рис. 6.

После добавления к проекту новых возможностей необходимо доработать тестбенч, чтобы их протестировать. Тестер должен проверять следующие возможности проекта: сброс (reset), счет (counting), сброс после превышения предела (correct roll-over operation), загрузка правильных данных (loading legal numbers), загрузка неправильных данных (loading illegal numbers) и продолжение сета после загрузки данных (correct counting after loading). Нет необходимости модифицировать тестер вручную — для этого используется уже готовая архитектура lab9.fc из элемента BCDRegister_Tester в библиотеке Lab_Templates. Скопируйте ее в ваш BCDRegister_Tester и назначьте используемой по умолчанию.

Теперь можно выделить BCDRegister_tb и запустить Tasks > Generate > Run Through Components. Однако попытка сгенерировать проект окажется неудачной, так как в скопированных архитектурах используется параметр настройки rollover, который еще не создан. Его надо создать вручную. Для этого необходимо открыть BCDRegister в редакторе диаграмм, щелкнуть правой кнопкой мыши по блоку BCDRegControl и выбрать пункт Object Properties, в появившемся окне выбрать закладку Generics и в появившейся таблице добавить rollover unsigned (7 dowto 0) «10011001». Обратите внимание, что значение «10011001» необходимо ввести в двойных кавычках. Далее необходимо открыть на редактирование BCDRegister_tb и аналогичным образом добавить параметр настройки rollover к блоку BCDRegister_tester.

Сгенерируйте и запустите на моделирование тестбенч BCDRegister_tb. Промоделируйте проект, убедитесь, что сброс в 0 после превышения значения 99 проходит нормально, а также убедитесь в правильности загрузки данных. Теперь необходимо изменить значение rollover и посмотреть, как проект будет работать в этом случае. Обратите внимание, что необходимо модифицировать rollover в двух местах: на BCDRegControl и на BCDRegister_tester. Есть два способа сделать это:

1. В процессе запуска сессии моделирования. Используйте меню Simulate > Start Simulation. В появившемся окне на вкладке Design выделите MyTestLib > BCDRegister_tb > struct, затем на вкладке Others при помощи кнопки Add добавьте параметр настройки rollover со значением 01011001 (не забудьте включить Override Instance-Specific Values) и нажмите Ok и еще раз Ok, чтобы запустить моделирование.
2. В окне Objects. Если окно Objects не открыто, откройте его через меню View > Objects. Затем во вложенном окне выберите Workspace, на вкладке sim выберите элемент BCDRegister_tb > struct. При этом в окне Objects, среди прочего, появится пункт rollover. Щелкните по нему правой кнопкой мыши и выберите Change.

Повторите процедуру модификации из пункта 2 для BCDRegControl. Способ, описанный в пункте 1, создает сразу оба значения rollover.

Отладьте проект, добейтесь того, чтобы BCDRegister сбрасывался в ноль после числа, заданного в rollover.

В следующей статье речь пойдет о расширенных возможностях синтеза.