

SystemView + Expressive = = HDL Design Studio

Эта статья знакомит читателя с возможностью по созданию описаний цифровых процессоров обработки сигнала (DSP) на языках программирования высокого уровня VHDL и Verilog, совместно предоставляемой новой версией программы SystemView 6.0 и программой Expressive.

Иосиф Златин

zlatin@pochta.ru

HDL Design Studio объединяет две очень мощных программы: SystemView и Expressive.

Программа Expressive упрощает создание, визуализацию и техническое обслуживание сложных иерархических проектов. К достоинствам программы Expressive можно отнести:

- Многоуровневые опции редактирования.
- Полную поддержку для сложных типов систем.
- Уникальную структуру интерфейса объектов.
- Поточный или классический схематический режимы ввода проекта.
- Универсальная поддержка от маленького до очень больших проектов (> 10 млн вентиляей).
- Упрощенная визуализация сложной информации.
- Инструментальные средства поиска специальных символов.

- Уникальные опции генерации кода.

Программа Expressive является инструментом управления проектами СВИС, оперирующими с проектами ASIC и FPGA. Эта программа также имеет средство создания мощного сценария (скрипта) размещения и автоматизации многих задач, требуемых в проекте, и, конечно, имеет генератор кода Verilog/VHDL, который читает файлы списка соединений SystemView и преобразовывает их в HDL.

HDL (Hardware Description Language) — язык описания аппаратуры, является формальной записью, которая используется на всех этапах разработки цифровых устройств. HDL Design Studio имеет дело с двумя разновидностями языка HDL: VHDL и Verilog.

Наиболее универсальным и распространенным языком описания аппаратуры является VHDL (Very high speed integrated circuits Hardware Description Language). VHDL фактически является международным стандартом в области автоматизации проектирования цифровых систем, это входной язык многих современных систем автоматизированного проектирования (САПР) как заказных, так и программируемых логических интегральных схем (ПЛИС). Язык VHDL предназначен для точного описания проектируемых систем и их моделирования на алгоритмическом и логическом этапе проектирования. С помощью VHDL можно моделировать цифровые схемы с учетом временных задержек. Наиболее распространенными САПР, использующими язык VHDL, являются программы MAX+Plus II и Quartus II (для ПЛИС фирмы Altera) и Foundation ISE и WebPACK ISE (для ПЛИС фирмы Xilinx). Для заказных СВИС могут быть использованы САПР фирмы Mentor Graphics: система моделирования ModelSim позволяет провести моделирование описаний, представленных на языке VHDL, система синтеза Leonardo Spectrum Level II позволяет получать по описаниям на языке VHDL схемы в заданных базах логических элементов.

VHDL позволяет описывать поведение, то есть алгоритмы функционирования цифровых систем, а также проводить иерархическое функционально-структурное описание систем, имеет средства для описания параллельных асинхронных процессов, регулярных структур и в то же время имеет все признаки языка программирования высокого уров-

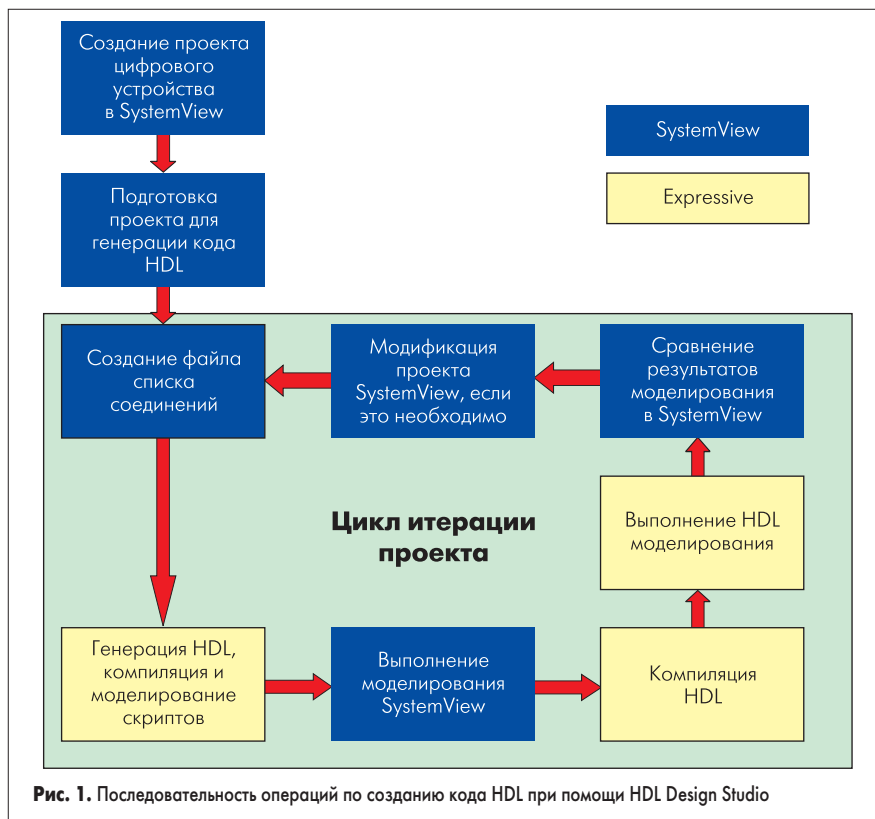


Рис. 1. Последовательность операций по созданию кода HDL при помощи HDL Design Studio

ня — позволяет создавать свои типы данных, имеет широкий набор арифметических и логических операций. Язык VHDL был разработан в 1987 году по инициативе министерства обороны США.

Язык описания аппаратуры Verilog был разработан фирмой Gateway Design Automation в 1984 году. В отличие от VHDL, структура и синтаксис которого напоминают язык АЛГОЛ, синтаксис Verilog напоминает язык С. В настоящее время ведущие пакеты синтеза систем на ПЛИС, такие, как продукты фирм Synopsys, Cadence, Mentor Graphics и большинства производителей поддерживают синтез с описаниями на языке Verilog.

На рис. 1 изображена схема, отображающая основные операции по созданию кода HDL с использованием HDL Design Studio.

Компиляция HDL и моделирование происходят автоматически под управлением скрипта. Новые результаты моделирования могут быть импортированы назад в SystemView для сравнения с другими результатами моделирования. Части проекта могут быть «черными ящиками», что означает, что генерация кода не выполнена; вместо этого существующий проект HDL замещен на модель SystemView. Эта особенность дает возможность постепенно преобразовывать проект в HDL.

Демонстрационная версия программного продукта HDL Design Studio доступна на сайте компании EnTegra, ее дистрибутив имеет размер 12 Мбайт. Пользоваться программой можно в течение 14 календарных дней после первого запуска. Лицензирование управляется файлом EntLicClient.exe, который появляется на панели задач Windows в виде пиктограммы замка:

Файл EntLicClient.exe находится в папке Program Files\Common Files\Entegra.

Для отображения информации о вашей лицензии (-й), дважды щелкните по пиктограмме замка.

EntLicClient отображает число дней для каждой лицензии программы. Когда лицензия демонстрационной версии истечет, можно связаться с вашим местным коммерческим офисом и запросить номер для продления лицензии. Чтобы сделать это, дважды щелкните по пиктограмме замка, выберите вашу программу, щелкнув правой кнопкой мыши, выберите «Set Expiry Date» («Установка окончания даты»). Появится диалоговое окно, показанное на рис. 2.



Рис. 2. Диалоговое окно Extend Evaluation Date

Позвоните дистрибьютору и сообщите ему 4 шестнадцатеричные числа из диалогового окна, показанного на рис. 2. Если дистрибьютор решит продлить лицензию демонстрационной версии, он сообщит вам десятичное число, которое вы должны ввести в поле «Enter the decimal number that your distributor gives you below», нажав затем кнопки Extend и Close. После этого будет отображен новый интервал действия демонстрационной версии.

С HDL Design Studio мы познакомимся с помощью различных примеров цифровых устройств, которые находятся в папке Program Files\SystemView\EnTegra\HDL Design Studio\Examples. В этих примерах мы будем использовать VHDL как разновидность языка HDL.

Умножитель-накопитель (multiply/accumulate)

Начнем знакомство с устройства «умножитель-накопитель» (multiply/accumulate).

Этот пример знакомит с основами HDL Design Studio. Вы изучите:

- как правильно подготовить проект DSP для генерации кода;
- как подсоединить и конфигурировать элемент «анализатор данных тестового кода» (TVSINK);
- как конфигурировать построитель скрипта;
- как выполнить генерацию кода HDL;
- как автоматически скомпилировать и промоделировать проект HDL;
- как импортировать и сравнить результаты моделирования.

Создание проекта

Запустим первый пример, загружая файл Example1\Example_1a.svu в SystemView. Схема примера показана на рис. 3.

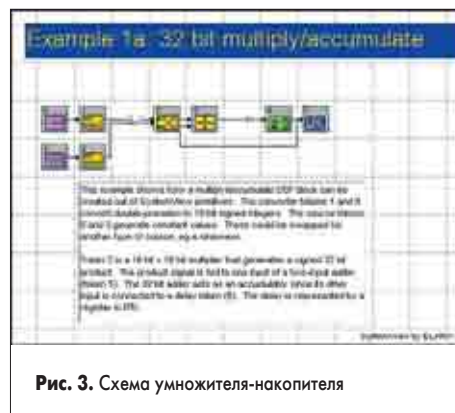


Рис. 3. Схема умножителя-накопителя

Этот пример представляет простой блок DSP, выполняющий операцию умножения-накопления и сформированный из DSP-элементов.

HDL Design Studio обычно использует двоичный дополнительный код. С помощью конвертеров (элементы 1 и 8) сигналы с двойной точностью (64 бита с плавающей запятой), поступающие с источников (0 и 3), конвертируются в 16-битный дополнительный двоичный код.

2-входовый умножитель (элемент 2) сконфигурирован как 32-разрядный умножитель величин со знаком. Умножитель в его чистой комби-

наторной логической форме — весьма сложная часть логики. В реальной микросхеме каждый логический вентиль обладает задержкой.

Задержка не должна быть больше установленной, и синхронизация является необходимым условием логических устройств. Для обеспечения законченного цикла синхронизации для логического умножителя вычисленный результат помещается в выходной регистр. Эта методика называется конвейерной обработкой.

Выходные данные умножителя (32-разрядное целое число со знаком) подаются на один из входов 2-входового сумматора (элемент 5).

В DSP-библиотеке в группе Arithmetic имеются два типа сумматора: простой комбинаторный логический тип и двухвходовый тип. Последний имеет параметр latency (время ожидания). Накопитель основан на комбинаторной версии с отдельным элементом задержки, действующим как регистр.

Активная задержка (элемент 6), связанная с выходом сумматора, выполняет две задачи. Сначала она действует как память, чтобы возратить предыдущую сумму к другому входу сумматора, таким образом, сумматор действует как накопитель. Во-вторых, она выполняет конвейерную функцию регистра, как описано выше.

Задержка элемента, когда он используется в петле обратной связи, должна всегда устанавливаться в режим «Active», чтобы гарантировать, что SystemView не вставит «невяные задержки» в нежелательной точке в системе с обратной связью.

Первый шаг в создании кода HDL заключается в объединении функциональных элементов в метасистему для создания иерархии. Ввиду того что языки VHDL и Verilog сильно зависят от иерархии проектов, генератор кода HDL Design Studio генерирует HDL только для метасистемы. Метасистемы могут, конечно, включать в свой состав другие метасистемы — таким образом можно получить многоуровневую иерархию.

Для создания метасистемы нажмите и не отпускайте правую кнопку мыши, движением мыши заключите в окно элементы 2, 5 и 6 (умножитель, сумматор и элемент задержки) и нажмите кнопку Create MetaSystem на инструментальной панели.

Схема проекта будет теперь выглядеть подобно показанной на рис. 4.

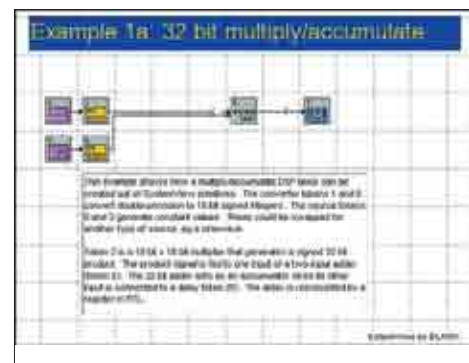


Рис. 4. Схема умножителя-накопителя с созданной метасистемой



Рис. 5. Диалоговое окно MetaSystem Name

Все иерархические элементы в HDL должны иметь название. Щелкните правой кнопкой по метасистеме и выберите «Custom Name» из выпадающего меню. В открывшемся диалоговом окне (рис. 5) введите название «MAC».

Это название будет транслировано в название объекта VHDL, который описывает проект.

Щелкните правой кнопкой мыши на метасистеме и выберите «View Meta-System» из открывшегося меню.

Вы можете видеть в окне метасистемы (рис. 6), что выбранные элементы теперь появились в метасистеме, и что добавились элементы MetaSystem I/O (вход-выход). В HDL входы и выходы называются портами. Все иерархические элементы должны иметь названия портов. Для того чтобы присвоить названия портам, щелкните правой кнопкой мыши по порту и выберите «Custom Name» из открывшегося меню. Заданное по умолчанию название портов полностью бессмысленно в проекте HDL. Сотрите его и введите новое название следующим образом:

- элемент10: InA
- элемент11: InB
- элемент12: Out.

Вы можете, конечно, назвать порты как вам нравится, но это соглашение о названиях удобно, потому что это — значения по умолчанию, используемые генератором тестовых сигналов.

Если порты I/O не переименованы, генератор списка соединений автоматически генерирует названия как:

- Inp0, Inp1, Inp2 ... и Out0, Out1, Out2 ...

Щелкните правой кнопки мыши на умножителе (элемент 2) и выберите «Edit Parameters». Появится диалоговое окно, показанное на рис. 7.



Рис. 7. Диалоговое окно задания параметров элемента Multiplier

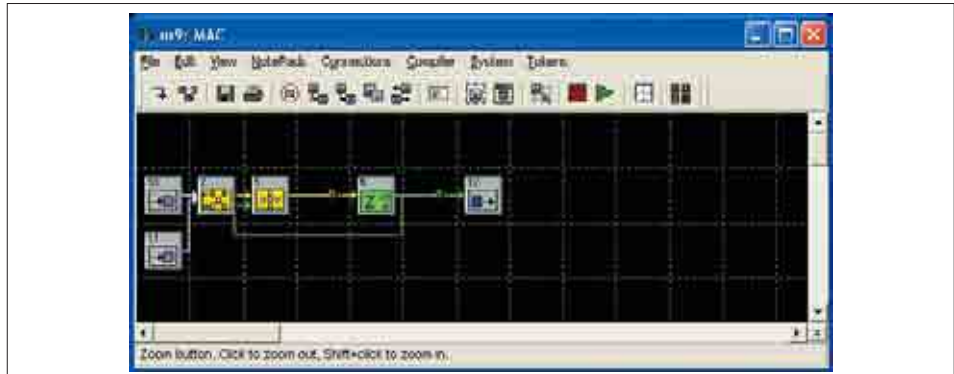


Рис. 6. Окно метасистемы

Установите параметр **latency** равным 1. Этот параметр моделирует задержку в 1 тактовый цикл в SystemView.

Теперь выполните моделирование, щелкнув по кнопке Run simulation на инструментальной панели диалогового окна System программы SystemView. Моделирование закончится чрезвычайно быстро. Теперь щелкните по кнопке Analysis tool окна Analysis. Для обновления результатов моделирования нажмите на небольшую всплывающую синюю кнопку в левой части инструментальной панели окна анализа Load New Sink Data и увидите график выходных данных схемы MAC (рис. 8).

Обратите внимание на задержку на один цикл перед первой выходной выборкой (0) — это из-за времени ожидания в умножителе.

Сохраните проект, возвратившись в область проекта (нажмите кнопку **System Window** на инструментальной панели окна анализа), затем выберите **File > Save** или нажмите **CTRL+S**.

Подготовка проекта для генерации кода HDL

Первым шагом в подготовке проекта для генерации кода является подключение эле-

мента TVIN, который используется для создания файлов тестовых векторов (входных сигналов). Элементы TVOUT также подключаются к любым сигналам, которые пользователь желает записать как выходные результаты. Эти специальные элементы распознаются генератором кода.

Перетащите элемент «Custom» с панели дополнительных библиотек в область проекта метасистемы и дважды щелкните по его пиктограмме. Появится диалоговое окно, показанное на рис. 9.

Выберите библиотеку «svu2hdl» как показано на рис. 9 (краткое описание данной библиотеки приведено в конце данной статьи), затем щелкните по элементу TVIN. Щелкните по левой кнопке «Parameters» для установки его параметров. Появится диалоговое окно, показанное на рис. 10.

Измените параметр «File path» («путь к файлу») на «InA.txt». Нажмите «OK», чтобы закрыть диалоговое окно, затем «OK» в диалоговом окне пользовательской библиотеки «svu2hdl», чтобы оставить элемент TVIN в области проекта. Затем элемент TVIN необходимо соединить со схемой. Подключите выход элемента 10 (входной порт InA) к входу нового элемента TVIN.

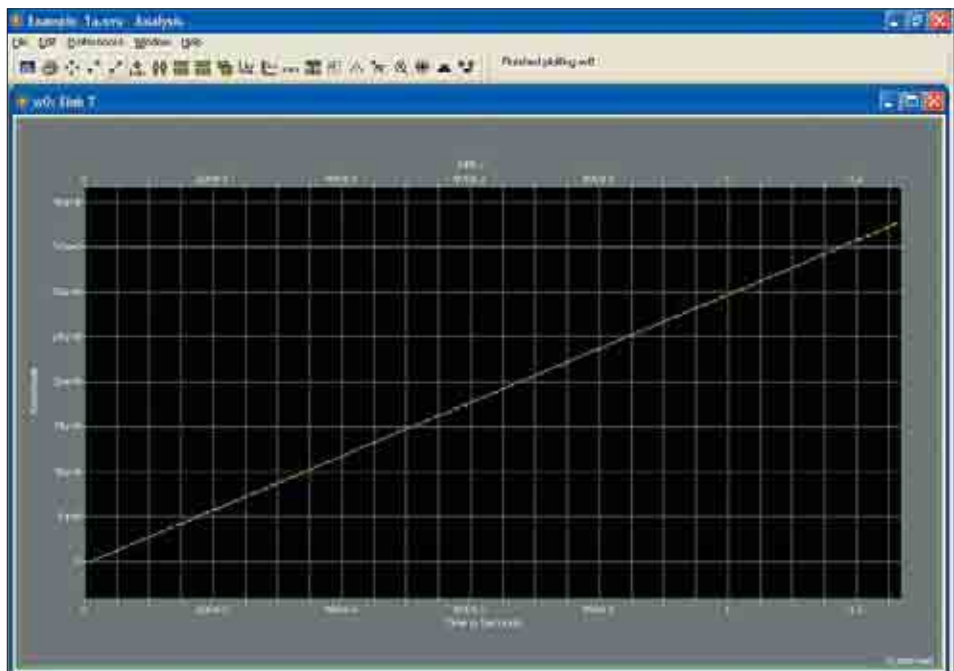


Рис. 8. Результаты моделирования схемы MAC в SystemView

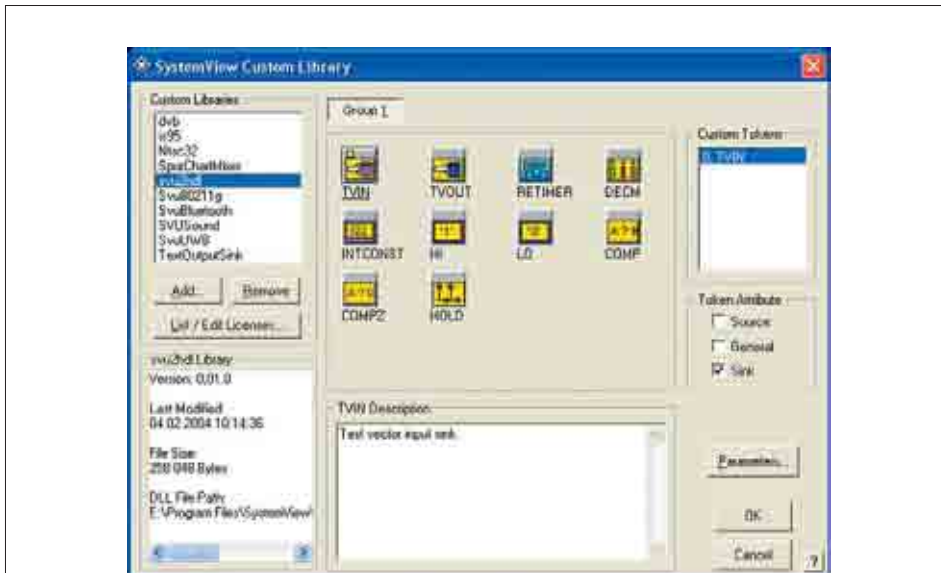


Рис. 9. Диалоговое окно пользовательской библиотеки svu2hdl

Продублируйте элемент TVIN и установите значение его параметра «File path» в «InB.txt». Подключите этот новый элемент к выходу элемента 11 (входной порт InB).

Вернитесь к диалоговому окну пользовательской библиотеки и создайте элемент TVOUT. Установите параметр «File path» в «Output.txt». Подключите его к элементу 6



Рис. 10. Диалоговое окно задания параметров элемента TVIN

Sample Delay. Проект метасистемы должен теперь выглядеть, как показано на рис. 11.

Теперь выполните моделирование SystemView, щелкнув на кнопке «Run». Это заставит элемент TVIN генерировать испытательные векторные файлы (входные сигналы), и элемент TVOUT создаст специальный текстовый файл (называемый plotlist.txt). Этот файл может использоваться скриптами и в особенности программой THT2PLT для получения файлов результатов моделирующего устройства.

Мы теперь готовы выполнить генерацию кода. Щелкните правой кнопкой мыши на метасистеме и выберите «Export HDL Design Studio...» из открывшегося меню. При этом запустятся инструментальные средства Expressive HDL Design Management (рис. 12).

Если у вас полная лицензия Expressive, вы можете теперь отредактировать проект в Expressive. Однако если у вас стандартная лицензия HDL Design Studio, то вы не сможете сделать никаких изменений в проекте, но сможете управлять моделированием и средствами генерации кода.

Проект MAC показан на рис. 12 в области Expressive GUI. Вы можете видеть три элемента, которые были транслированы генератором кода в компоненты VHDL. Двойной щелчок на них отобразит лежащий в их основе код VHDL.

На инструментальной панели Expressive размещается кнопка Code Generation. Щелкните по кнопке Code Generation, и появится диалоговое окно, показанное на рис. 13.

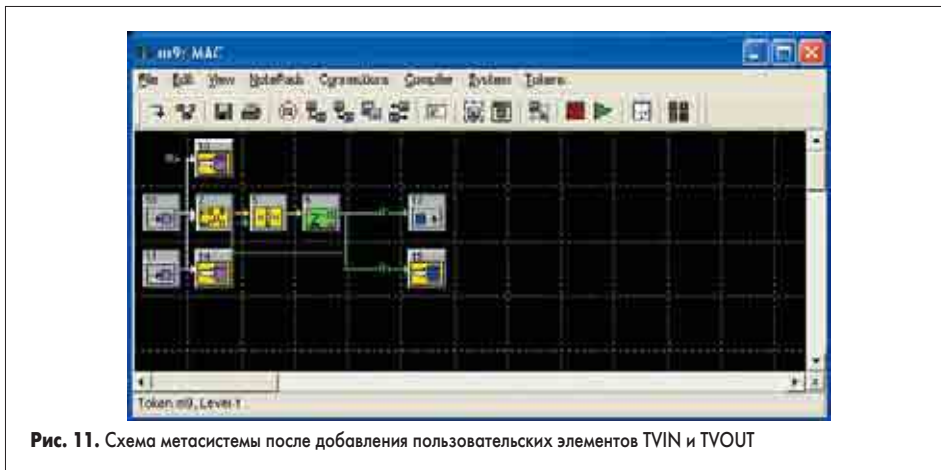


Рис. 11. Схема метасистемы после добавления пользовательских элементов TVIN и TVOUT

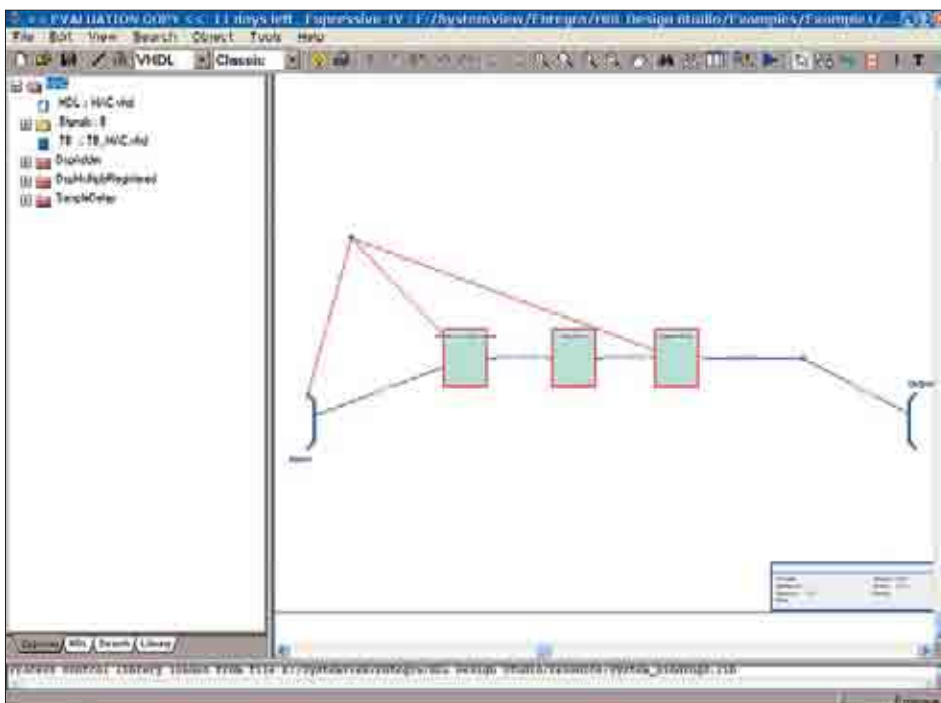


Рис. 12. Диалоговое окно Expressive HDL Design Management



Рис. 13. Диалоговое окно HDL Code Generation

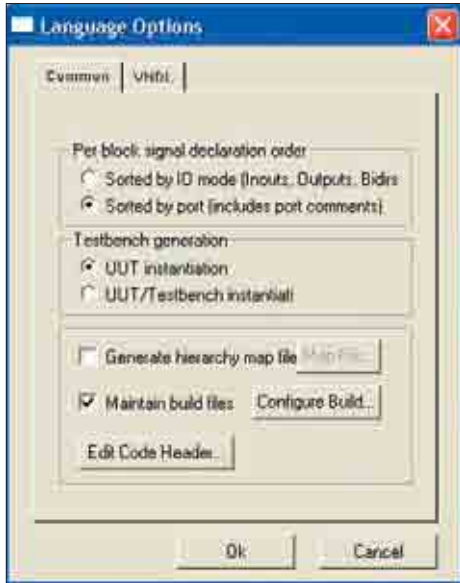


Рис. 14. Диалоговое окно Language Options (в демонстрационной версии отсутствует закладка Verilog)

Перед генерацией кода нажмите на кнопку HDL Options, чтобы открыть диалоговое окно, показанное на рис. 14.

Теперь нажмите на «Configure Build» для открытия диалогового окна Build Editor (построение конфигурации), показанного на рис. 15.

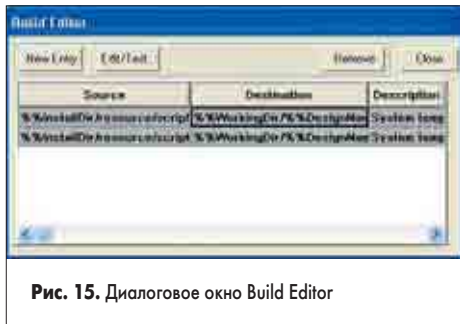


Рис. 15. Диалоговое окно Build Editor

Это диалоговое окно позволяет пользователю конфигурировать автоматическое создание скрипта. Expressive's Automatic Script Generator (ASG) дает возможность автоматически сгенерировать много различных типов скриптов, используя набор макрокоманд. Скрипты могут быть: ModelSim, файлы .DO, файлы DOS Batch, скрипты TCL, Make-файлы (сборочный файл проекта, отражающий все взаимосвязи между исходными файлами, установками среды, параметрами компиляции и т. д.) или определяемые пользователем скрипты. Скрипты сохраняются как шаблон файлов (.TEM) в Program Files/ExpressiveSystems/Resource/Scripts.

Когда пользователь конфигурирует Build (построение), используя диалоговое окно «Maintain Build files», шаблоны скриптов, перечисленные в столбце Source, считаются макрорасширениями для ASG. Результирующие расширенные скрипты помещаются в целевой каталог.

Для рассматриваемого примера, описанного в данной статье, простой скрипт для компиляции и моделирования VHDL использует моделирующее устройство ModelSim. Предусмотренный для этого скрипт mtisim.tem показан ниже:

```
##
## Default Model Technologies simulation script
##
cd «%%CodeGenDir»
vlib work
vmap work work
%%ForAllBlocks(vcom -work work «%%HdlFile»)
vcom -work work -93 «%%CodeGenDir%%TbFileName%%HdlExt»
vsm work.%%TbInstName
view wave
add wave %%TbInstName/*
run -all
##quit -f
```

Макрокоманды отмечены символом «%%». Этот файл расширен с помощью ASG в файл mtisim.do, показанный ниже:

```
##
## Default Model Technologies simulation script
##
cd «E:/Program Files/SystemView/EnTegra/HDL Design Studio/Examples/Example1/»
vlib work
vmap work work
vcom -work work «E:/Program Files/Expressive Systems/resource/svu/hdlLibrary/SampleDelay.vhd»
vcom -work work «E:/Program Files/Expressive Systems/resource/svu/hdlLibrary/DspAdder.vhd»
vcom -work work «E:/Program Files/Expressive Systems/resource/svu/hdlLibrary/DspMultiply.vhd»
vcom -work work «E:/Program Files/SystemView/EnTegra/HDL Design Studio/Examples/Example1/MAC.vhd»
vcom -work work -93 «E:/Program Files/SystemView/EnTegra/HDL Design Studio/Examples/Example1/TB_MAC.vhd»
vsm work.TB_MAC
view wave
add wave TB_MAC/*
run -all
##quit -f
```

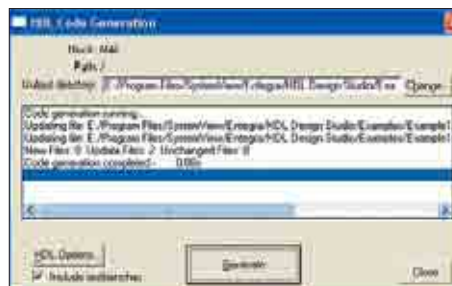


Рис. 16. Диалоговое окно HDL Code Generation после генерации кода

Этот скрипт выполняется Expressive автоматически, когда пользователь моделирует проект. В настоящее время Expressive сконфигурирована для выполнения моделирующим устройством ModelSim, но эта конфигурация может быть изменена на использо-

вание обязательного редактора обработки Expressive.

Нажмите «OK» в редакторе построения конфигурации и в окне «Language Options», затем нажмите кнопку «Generate». После генерации кода диалоговое окно будет выглядеть, как показано на рис. 16.

Будут сгенерированы два файла VHDL:

TB_MAC.vhd — испытательные сигналы; MAC.vhd — блок умножитель-накопитель.

Обратите внимание, название MAC совпадает с названием метасистемы. Генератор кода только генерирует код для файлов, которые должны быть модифицированы. Если вы уже запустили генератор кода и не произвели никаких изменений параметров проекта, файлы заново создаваться не будут.

Закройте диалоговое окно генерации кода. Мы теперь готовы выполнить моделирование.

Перед выполнением моделирования необходимо сохранить проект SystemView, нажав CTRL+S в SystemView.

Выполнение моделирования

Прежде чем начать выполнение моделирования HDL, необходимо произвести моделирование SystemView для генерации тестовых векторов. Запустите моделирование, щелкнув в SystemView на кнопку Run.

В Expressive нажмите на кнопку Simulation. Эта кнопка объединяет шаги генерации кода и моделирования, потому что обычно код HDL должен быть на соответствующем уровне прежде, чем он будет откомпилирован для моделирования.

После нажатия кнопки Simulation запустится файл mtisim.do, который был создан Automatic Script Generator. Этот скрипт компилирует файлы VHDL, после чего запускается моделирующее устройство. Появится диалоговое окно, которое показано на рис. 17.

Не должно вызывать беспокойство сообщение «Fatal Error!» («фатальная ошибка»). На самом деле это не является действительной ошибкой — скрипт заставляет ModelSim работать, но программа не может выполняться дальше, потому что она достигла конца одного из файлов тестовых векторов. Поэтому моделирующее устройство останавливается и выдает сообщение об ошибке.

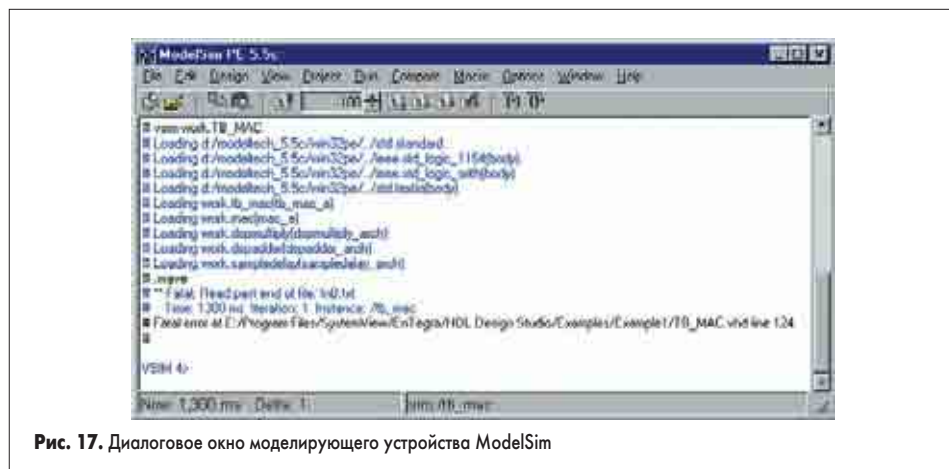


Рис. 17. Диалоговое окно моделирующего устройства ModelSim

Предполагая, что моделирующее устройство прекратило работать, возвратитесь в SystemView и активизируйте инструментальные средства анализа, щелкнув по кнопке Analysis Tool.

На инструментальной панели окна Analysis выберите File > Open SystemView plot... Выберите файл Output.PLT, который является двоично отформатированной версией файла Output.txt, созданного моделирующим устройством.

Наложите результат моделирования SystemView на результат моделирования HDL, щелкните левой кнопкой мыши на граничной области одного из графиков, затем перетащите один график поверх другого. Вы должны убедиться, что HDL и моделирование SystemView точно совпадают.

Альтернативный способ импортировать файл результата моделирования HDL заключается в выборе File > Import External Data... > Open New Plot Window. При этом отобразится диалоговое окно, показанное на рис. 18.



Рис. 18. Диалоговое окно Import External Data

Нажмите на кнопку «Select File...», оставляя установку Data Format «Text».

Выберите файл Output.txt. В появившемся диалоговом окне (рис. 19) SystemView попросит вас определить, какой столбец данных задает данные по оси X и какой — по оси Y.



Рис. 19. Диалоговое окно позволяет выбрать первый столбец данных в качестве данных, задаваемых по оси X или по оси Y

Щелкните по кнопке X-Axis. Результаты моделирования появятся в окне графика.

Синтез проекта

Код VHDL (или Verilog), создаваемый генератором HDL, является годным для синтеза.

Следующий пример показывает результаты синтеза для микросхемы FPGA XC2V40 фирмы Xilinx, использующей программу Leonardo Spectrum II.

Эти три блока метасистемы, несомненно, могут быть замечены в схеме после синтеза.

Обратите внимание, что генератор кода автоматически добавил синхронизацию и асинхронный сброс.

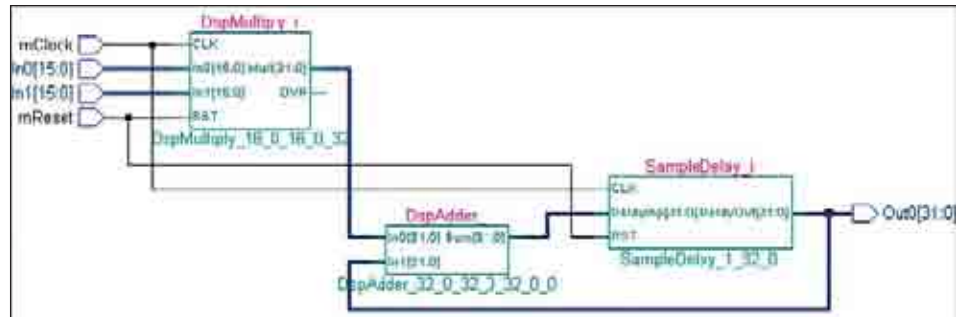


Рис. 20. Результаты синтеза проекта фильтра CIC в программе Leonardo Spectrum II

Каскадный интегрирующий гребенчатый фильтр (A Cascaded-Integrator Comb (CIC) Filter)

Этот пример знакомит читателя с некоторыми из наиболее продвинутых особенностей HDL Design Studio:

- как выполняется каскадный интегрирующий гребенчатый фильтр (CIC) в SystemView;
- как SystemView и HDL Design Studio осуществляют прореживание,

Трехсекционный (3-го порядка) каскадный интегрирующий гребенчатый фильтр (CIC) обеспечивает следующую передаточную функцию:

$$H(z) = [(1-z^{-4})/(1-z^{-1})]^3$$

Создание проекта

Загрузите проект Examples\Example2\Example_2a.svu в SystemView. Схема проекта показана на рис. 21.

Вы можете ясно видеть три каскада интегратора, прореживатель, затем три каскада дифференциатора вышеупомянутой схемы. Элементы сумматоров 50, 51 и 52 вместе с задержками обратной связи формируют три каскада интегрирования. Каждый сумматор

имеет разрядность 21 бит, входная разрядность — 15 бит, и N+2S бит требуется для арифметических операций. S является 3, следовательно, требование 15 + 2 × 3 = 21 бит. Вычитающее устройство (элементы 45, 46 и 47) выполняет дифференцирование и имеет ту же разрядность 21 бит. Элементы 31, 32 и 33 выполняют округление, преобразовывая 21 бит на выходе элемента 47 до 15 бит.

Элемент 19 («прореживатель») является уникальным элементом. Он относится к классу элементов, называемых «Time active». Это означает, что элемент изменяет действующую частоту дискретизации. Прореживатель установлен с коэффициентом прореживания 4. Это означает, что отсчеты x(0), x(3), x(7), x(11) и т. д. сохраняются, а другие отсчеты отбрасываются.

Регистры конвейерной обработки должны быть расположены между большими блоками комбинаторной логики типа сумматора.

Как HDL Design Studio выполняет многоуровневые иерархические проекты

Сначала установите проект для генерации кода и выполните последовательность операций по генерации кода как описано в преды-

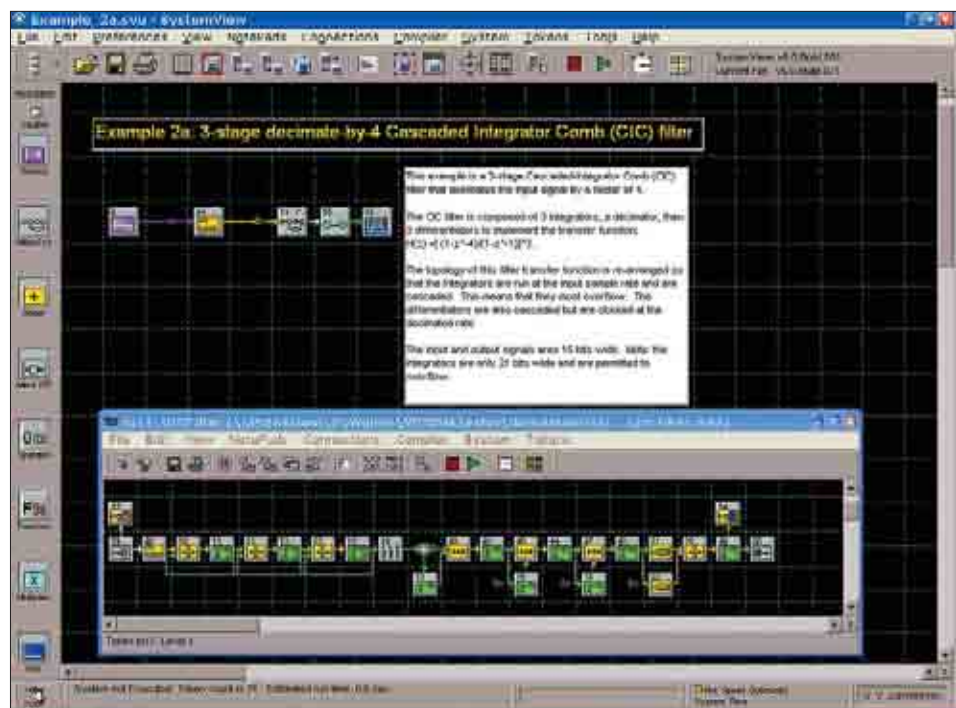


Рис. 21. Схема каскадного интегрирующего фильтра

дущем примере в разделе «Подготовка проекта для генерации кода HDL». Теперь взгляните на описание объекта для компонента CICFilter в файле VHDL CICFilter.vhd.

```
-- Entity_Start --->>>
---

library IEEE;
use IEEE.std_logic_1164.all;

entity CICFilter is
    port( mClock : in std_logic;
          mReset : in std_logic;

    -- Port: Inputs

    CE : in std_logic;
    In0 : in std_logic_vector(14 downto 0);

    -- Port: Outputs

    Out0 : out std_logic_vector(14 downto 0);
    Token_59_RDY : out std_logic
    );
end CICFilter;
---
--- Entity_End <<<---
```

Здесь имеется сигнал CE (Clock Enable). Объект, конечно, имеет вход (In0), выход (Out0) и сигналы синхронизации и сброса, как было в предыдущем примере MAC. Однако теперь есть еще дополнительный сигнал RDY. Хотя CICFilter синхронизирован тактовой частотой системы, выходные данные имеют прореженную частоту. Сигнал RDY указывает прореженные выходные отсчеты.

Теперь посмотрите на описание для элемента «прореживатель»:

```
component Decimate
    generic( Factor : integer := 4;
             RegisterLength : integer := 21
            );
    port( CLK : in std_logic;
          ST : in std_logic;

    -- Port: Inputs

    CE : in std_logic;
    n0 : in std_logic_vector(RegisterLength-1 downto 0);

    -- Port: Outputs

    Out0 : out std_logic_vector(RegisterLength-1 downto 0);
    RDY : out std_logic
    );
end component;
```

Decimator (прореживатель) является элементом, который имеет вход CE и выход RDY. Прореживатель имеет счетчик, который осуществляет счет всякий раз, когда на вход CE подается логическая единица. Когда счетчик достигает Factor — 1 (коэффициент = -1; коэффициент = 4 в случае прореживания с коэффициентом прореживания = -4), входной отсчет фиксируется триггером. Сигнал RDY устанавливается в единицу для одного тактового цикла. Сигнал RDY указывает для следующего блока (или блоков), что тракт данных построен правильно.

Как показано в коде VHDL, выход прореживателя RDY соединен с сигналом, называемым RDY_buf:

```
-- Externally sourced HDL file
-- File: E:/Program Files/SystemView/Entegra/HDL Design
Studio/resource/svu/hdlLibrary/Decimate.vhd

Decimate_i : Decimate generic map( Factor => 4,
                                   RegisterLength => 21 )


    port map( CLK => mClock,
              RST => mReset,
              CE => CE,
              In0 => Token_25_DelayOut,
              Out0 => Token_59_Out0,
              RDY => Token_59_RDY_buf );
```

Моделирование фильтра CIC

Промоделируйте пример фильтра CIC (Example_2a), щелкая по кнопке Simulation run. Элемент TVSINK автоматически компилирует код VHDL, затем, после того как моделирование SystemView завершится, выполняется моделирование ModelSim.

Переключитесь на инструментальные средства анализа и импортируйте результаты моделирования SvUOut0.PLT и Out0.PLT.

Вы увидите, что выходные сигналы точно не совпадают. Причиной этого является то, что в SystemView элемент decimation (прореживатель) не обеспечивает правильную генерацию кода VHDL.

Среди пользовательских библиотек выберите библиотеку svu2hdl. Замените элемент decimation на элемент DEC . Установите параметр Decimation factor (коэффициент прореживания) равным 4.

Этот элемент правильно моделирует VHDL, при этом элемент «прореживатель» вносит задержку на один тактовый цикл.

Теперь выполните генерацию кода и моделирование. Вы теперь увидите, что результаты совпадают. Этот пример сохранен как Example_2b.svu.

Окончание следует

Литература

1. Разевиг В. Д., Лаврентьев Г. В., Златин И. Л. SystemView — средство системного проектирования радиоэлектронных устройств / Под ред. В. Д. Разевига. М.: Горячая линия-Телеком. 2002.
2. Златин И. Новые возможности SystemView // Компоненты и технологии. 2003. № 1.
3. Разевиг В. Д., Златин И. Л. Новые возможности SystemView // EDA Express M., Издательство ОАО «Родник Софт». 2003. № 7.
4. Златин И. Пользовательские библиотеки (Custom Library) и многостанционный доступ с кодовым разделением каналов (CDMA) в SystemView // Компоненты и технологии. 2003. № 8.
5. Златин И. Пользовательская библиотека WNL 80211g в SystemView // Компоненты и технологии. 2003. № 9.
6. Златин И., Кадышев С. SystemView + Matlab + Simulink // Компоненты и технологии. 2004. № 2.
7. Златин И. Еще раз о пользовательских библиотеках SystemView // Компоненты и технологии. 2004. № 3.
8. Златин И. Цифровое телевидение (DVB) и технология беспроводной сверхширокополосной связи (UWB) в SystemView // Компоненты и технологии. 2004. № 4.
9. Стешенко В. Б. EDA. Практика автоматизированного проектирования радиоэлектронных устройств. М.: Нолидж. 2002.
10. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPACK ISE. М.: Горячая линия-Телеком. 2003.
11. Adrian Nash — HDL Design Studio. EnTegra, Inc. 2003.