УДК 004.4.233

НОВОЕ СРЕДСТВО ВЕРИФИКАЦИИ ПЛИС ДЛЯ ЗАДАЧ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

Романов А.М., *к.т.н., научный сотрудник кафедры проблем управления Института кибернетики МИРЭА, e-mail: AlexRomashka@yandex.ru;*
Слащёв Б.В., *программист кафедры проблем управления Института кибернетики МИРЭА, e-mail: slaschov.bogdan@yandex.ru.*

NOVEL FPGA VERIFICATION TOOL FOR DIGITAL SIGNAL PROCESSING APPLICATIONS

Romanov A.M., Slashev B.V.

The article introduces a novel software tool Vmodel toolbox, that allows to increase FPGA verification performance in digital signal processing applications. The Vmodel toolbox specialtity is the ability to simulate FPGA as a part of Simulink model without external HDL-simulators. The concept, functionality and key advantages are described. The results of successful implementation in FPGA design projects are described.

Key words: FPGA, verification, digital signal processing, Vmodel toolbox, HDL-simulators.

Ключевые слова: ПЛИС, верификация, цифровая обработка сигналов, Vmodel toolbox, HDL-симулятор.

Предлагается новое программное средство Vmodel toolbox, позволяющие существенно повысить производительность верификации ПЛИС в задачах цифровой обработки сигналов. Особенностью Vmodel toolbox является возможность моделирования ПЛИС как части Simulink модели без использования внешних HDL-симуляторов. Описывается принцип работы, функциональность и основные достоинства нового программного обеспечения. Приводятся примеры удачного внедрения в реальных проектах.

Введение

На заре применения программируемых логических интегральных схем (ПЛИС) их основным назначением была реализация нестандартных логических преобразований, для которых выпуск отдельных микросхем был экономически не выгоден. После, к этому добавилась потребность в реализации логики для сопряжения нескольких различных интерфейсов между собой [1]. Эти задачи определили основные средства отладки, которые присутствуют в современных пакетах моделирования аппаратно-программного обеспечения (АПО) ПЛИС: временные диаграммы изменения сигналов, временные диаграммы изменения отдельных битов сигнала, возможность просмотра состояния сигналов в бинарном, восьмеричном, десятичном и шестнадцатеричном виде и т.д. Тесты для модулей ПЛИС, как правило, пишутся на одном из HDL-языков, которые идеально подходят для описания дискретных логических преобразований происходящих при обработке того или иного цифрового интерфейса. С развитием технологии ПЛИС и появлением ресурсоемких микросхем основной вектор их применения начал смещаться в сторону сложной цифровой обработки сигналов (ЦОС), однако способы отладки, представленные в современных средствах моделирования, не претерпели существенных изменений. В тоже время потребности разработчика ЦОС на базе ПЛИС в инструментах проектирования и отладки существенно отличают от аналогичных потребностей разработчика интерфейсной логики. Начать стоит с того, что первый этап проектирования блоков ЦОС проходит в пакетах математического моделирования (MATLAB, SciLab и т.д.). Стандартом на данный момент де-факто является MATLAB. На этом этапе еще до описания на HDL проверяется верность самого математического преобразования, которое будет реализовано на ПЛИС, и соответствие его требуемым характеристикам. Для этого доступно множество инструментов. Так, в пакете Simulink легко можно смоделировать цифровой фильтр произвольной конфигурации, подать на него сигнал с заданными спектральными характеристиками, а потом убедиться в том, что спектр выходного сигнала удовлетворяет требования спецификации. На следующем этапе проектирования происходит реализация синтезированного ранее блока ЦОС на одном из HDL-языков, а для отладки полученного описания приходится использовать HDL-симуляторы, инструментарий которых заточен под отладку интерфейсной логики. В результате для получения в процессе верификации входных сигналов с заданными спектральными характеристиками требуется описание процесса их формирования на HDL-языке, что сопоставимо по трудозатратам с созданием описания самого модуля ЦОС. В дополнение к этому такой тестовый HDL-модуль потребует отдельного тестирования. Можно сформировать требуемые сигналы при помощи MATLAB и загрузить их в HDL-симулятор из файла, но такой подход вряд ли можно считать удобным. Аналогично обстоят дела с анализом результатов. В ЦОС практически не востребован побитный анализ отдельных сигналов или представление сигналов в двоичной или восьмеричной форме. В тоже время часто применяется представление сигналов в

форме дробных чисел с фиксированной или плавающей точкой, которая не может быть корректно отражена на временных диаграммах в большинстве средств моделирования. Так же в HDL-симуляторах отсутствуют возможность построения многокоординатных графиков и поверхностей, средства частотного анализа сигналов и т.д. В конечном счете результаты моделирования блока ЦОС приходится сохранять и анализировать в MATLAB или аналогичных пакетах.

Понимая недостаточность имеющегося функционала для успешной отладки модулей ЦОС, разработчики наиболее продвинутых HDL-симуляторов добавили в них специализированный режим со-симуляции с MATLAB. В этом режиме HDL код моделируется в специализированном симуляторе, а остальные элементы системы моделируются в Simulink. Связь двух программ между собой осуществляется по сетевому протоколу TCP/IP непосредственно в процессе моделирования. Такой подход оказался особенно востребован при реализации ЦОС в системах управления, когда для верификации необходимо моделировать функционирование АПО ПЛИС совместно со сложными динамическими аналоговыми объектами, которые описываются дифференциальными уравнениями высокого порядка, а на входы блока ЦОС помимо управляющих сигналов поступают сигналы обратной связи от объекта управления. Один из примеров успешного применения данного подхода при создании нелинейной системы управления двигателем на базе ПЛИС описан в [2]. Главным недостатком со-симуляции является низкая скорость моделирования, связанная с постоянным обменом данными между разными программами. Помимо этого сохраняется, хоть и в меньшей степени, необходимость работы в ходе верификации с двумя различными программами, а также необходимость приобретения отдельных лицензий для каждой из них.

Для того, чтобы исключить использование внешнего HDL-симулятора в компании MathWorks предложили оригинальный подход к аппаратной реализации ЦОС на базе ПЛИС с использованием их продукта HDL Coder, который позволяет автоматически без участия разработчика создавать HDL описание блоков Simulink на языках Verilog и VHDL. При использовании HDL Coder разработчик работает только с моделью, а все этапы её преобразования в файл конфигурации ПЛИС проходят автоматически. Это позволяет в простых проектах обходиться вообще без моделирования HDL кода, ограничиваясь только отладкой функциональной модели в Simulink. Однако в сложных проектах такой подход не применим, так как степень оптимизации кода, созданного при помощи HDL Coder, на сегодняшний день уступает коду профессионального разработчика. И если на стадии прототипирования это не так важно, то в серийных изделиях, где размер АПО ПЛИС напрямую влияет на стоимость и характеристики будущего устройства, HDL-код пока приходится писать вручную. Пример успешного применения HDL Coder для решения задач ЦОС описан в [3].

При разработке сложных устройств на базе ПЛИС активно используются готовые модули (IP-ядра), которые были созданы ранее в ходе других проектов или куплены у сторонних разработчиков. Это позволяет существенно снизить трудозатраты и сократить сроки выхода на рынок. Для использования стороннего (не созданного автоматически) HDL кода, в HDL Coder есть механизм подключения внешних IP-ядер в формате «черного ящика». Суть его заключается в том, что разработчик самостоятельно создает функциональную модель IP-ядра на Simulink, которой ставит в соответствие имеющийся у него HDL код. В ходе верификации используется именно функциональная модель, а в момент генерации кода HDL Coder автоматически использует код соответствующего ей IP-ядра. На практике такой подход имеет существенные недостатки. Помимо дополнительных трудозатрат на создание функциональной модели для каждого подключаемого IP-ядра, использование «черных ящиков» в HDL Coder создает опасность того, что какие-то из особенностей функционирования IP-ядра не будут учтены в замещающей его модели. Это может привести к тому, что функционирование HDL кода, синтезированного в ПЛИС, будет отличаться от результатов, полученных в ходе верификации. Поэтому для проверки соответствия разработанной функциональной модели используемому IP-ядру нельзя ограничиться использованием MATLAB и приходится применять специализированные HDL-симуляторы.

Описанные выше недостатки существующих программных пакетов для верификации АПО ПЛИС обосновывают актуальность создания нового инструмента, который бы был более приспособлен для задач ЦОС. На основании имеющегося опыта разработки блоков ЦОС для ПЛИС можно сформулировать следующие требования к новому программному обеспечению:

1) высокая скорость моделирования HDL-кода;

2) кроссплатформенность;

3) наличие возможности:

– проводить верификацию из интерфейса MATLAB;

– формировать тестовые воздействия средствами MATLAB;

– проводить анализ моделирования средствами MATLAB;

– проводить автоматическую верификацию без участия человека;

– на основе HDL-кода автоматически создавать блок Simulink;

– использовать полученные Simulink модели на компьютерах без специализированных средств моделирования HDL кода;

– автоматически создавать модели типа «черный ящик» для HDL Coder;

4) отсутствие необходимости покупать дополнительные лицензии, кроме MATLAB;

Всем этим требованиям удовлетворяет новый программный пакет Vmodel toolbox, созданные на кафедре проблем управления Института кибернетики МИРЭА [4].

Принцип работы Vmodel toolbox

При разработке данного инструмента моделирования сразу был отвергнут вариант использования внешнего симулятора, который бы связывался в процессе моде-

лирования с MATLAB. По сути это бы являлось попыткой повторить режим со-симуляции, производительность которого на примере коммерческих симуляторов оказалась недостаточной. Вместо этого был выбран путь создания моделей в виде прекомпелированных библиотек MATLAB. Такой подход имеет сразу несколько достоинств: в процессе компиляции код модели оптимизируется, что увеличивает производительность симуляции; обмен между MATLAB и моделью происходит через оперативную память, что также сказывается на быстродействии; созданную модель можно запустить на любом компьютере с MATLAB, что позволяет передать её специалистам работающим над проектом, но не занятых в проектировании АПО ПЛИС.

Основой разработанного инструмента стал симулятор с открытым кодом Verilator [5]. Он позволяет на основе HDL-кода на языке Verilog создавать поведенческие модели на языках C++ или SystemC, выполненные в виде объектов. Эти модели могут быть впоследствии скомпилированы и подключены к внешнему программному обеспечению. Verilator нашел широкое применение при разработке микропроцессорных системы. По результатам сравнения с другими симуляторами отмечается, что скорость моделирования моделей, созданных в Verilator, выше в несколько раз [6, 7].

Алгоритм работы Vmodel toolbox представлен на рис. 1. Пользователь передает в программу HDL-код, который необходимо промоделировать, и конфигурацию будущей модели. Конфигурация включает в себя данные о расположении файлов проекта, типах моделей, которые надо создать, параметрах тактовых сигналов и т.д.

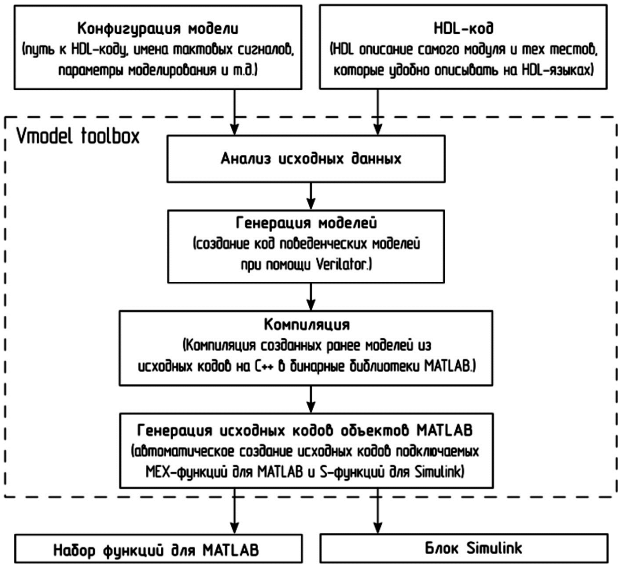

Рис. 1. Алгоритм работы пакета Vmodel toolbox

На основании заданной конфигурации Vmodel toolbox автоматически генерирует исходные коды объектов MATLAB на языке C++, подключает к ним поведенческие модели АПО ПЛИС, которые генерируются при помощи Verilator, и компилирует их в бинарные MEX-файлы и S-функции для MATLAB и Simulink соответственно. В дальнейшем пользователь использует в процессе моделирования только скомпилированные файлы, никак не задействуя Verilator.

Процедура установки

Vmodel toolbox свободно распространяется с открытым кодом по лицензии LGPL[8]. Дистрибутив может быть скачан с официального сайта проекта [9]. Vmodel toolbox требует для своей работы установленный пакет MATLAB с дополнением Simulink версии 2011a или старше. Никакие другие дополнения не требуются. Установка производится автоматически путем запуска скрипта install_vmodel.m из MATLAB. Если на компьютере не был установлен Verilator, то он будет автоматически установлен с официального сайта. На сегодняшний день известно об успешном использовании Vmodel toolbox в операционных системах Windows 7, Windows 8, Ubuntu Linux 12.04 LTS, Ubuntu Linux 14.04 LTS. Потенциально программное обеспечение Vmodel toolbox должно работать на любой операционной системе, которая позволяет запустить MATLAB.

Конфигурация модели

Для конфигурации модели необходимо создать в MATLAB структуру конфигурации и передать её функции vmodel(), которая станет доступна в пакете MATLAB после установки Vmodel toolbox. Основные параметры конфигурации представлены в табл. 1.

При использовании нескольких тактовых сигналов для каждого из них можно задать частоту, фазовых сдвиг и величину джиттера. В процессе создания модели эти величины пересчитываются относительно указанной в конфигурации частоты ПЛИС. Таким образом если, например, в модели Simulink поднять частоту ПЛИС в 2 раза, то и все внутренние частоты возрастут в 2 раза так, как если бы они были сформированы при помощи блока синтеза частоты в ПЛИС.

Одной из уникальных возможностей Vmodel toolbox является моделирование случайных отказов. Достаточно задать требуемую вероятность отказа в параметр rchg_probability при создании модели, и с этой вероятностью будут моделироваться случайные изменения сигналов, выходов регистров и элементов памяти. Данная функция оказывается крайне полезной при разработке блоков ЦОС для систем с высокими требованиями к отказоустойчивости.

Для оценки репрезентативности тестовой выборки в Vmodel toolbox реализована поддержка подсчета покрытия кода тестами, которая доступна как при моделировании в Simulink, так и в MATLAB. Благодаря имеющемуся функционалу, пользователь получает информацию о том, сколько с момента начала моделирования было вхождений в ту или иную строчку HDL-кода. При этом существует возможность гибкой настройки того, какие модули участвуют в анализе, какое количество вхождений является минимально-допустимым и какие результаты анализа будут доступны на выходе.

Так как параметры HDL модулей не могут быть изменены в ходе их работы, они должны быть заданы до компиляции модели. Это можно сделать как в ручную, изменив исходный код модуля, так и автоматически при помощи функции insert_with_params, которая создает Verilog файл, содержащий экземпляр указанного модуля с заданными из M-языка параметрами. Это открывает возможность автоматизации процесса верификации параметризированных IP-ядер.

Таблица 1. Параметры конфигурации модели Vmodel toolbox

Параметр	Описание
src_filename	Путь к исходному коду главного HDL модуля
Dirlist	Список директорий, где необходимо искать исходные коды
Output	Директория, куда будут помещены созданные поведенческие модели
no_matlab_model	Не создавать модель для MATLAB
no_simulink_model	Не создавать модель для Simulink
constr_name	Имя функции конструктора
sim_name	Имя функции моделирования заданного периода времени
sim_name2	Имя функции моделирования до заданного условия
break_condition	Условие остановки на языке C++ для функции моделирования до заданного условия
fpga_freq	Частота ПЛИС
sample_time	Период расчета модели Simulink
clk_name	Имя тактового сигнала в режиме с одним тактовым сигналом
multiclock	Включить режим с несколькими тактовыми сигналами
Clocks	Структура описания тактовых сигналов в режиме с несколькими тактовыми сигналами
random_seed	Положительное 32-битное число, которым будут проинициализированы генераторы случайных чисел, которые используются при моделировании джиттера и случайных отказов. Параметр необходим для обеспечения повторяемости экспериментов
Signals	Параметр определяет какие сигналы будут отображаться пользователю по результатам моделирования: 'all' – все доступные сигналы; 'top' – только выходы главного модуля; 'public' – выходы главного модуля и сигналы, помеченные в коде комментарием /*verilator public*/; 'blackbox' – аналог 'top', но создается «черный ящик» для HDL Coder
clk_to_out	Добавить состояние тактовых генераторов в результаты моделирования
inputs_to_out	Добавить состояние входов в результаты моделирования
reinterpret_floats	Параметр определяет будут ли значения переданные из MATLAB в формате с плавающей точкой округлены и присвоены как целые числа или побитно присвоены в формате IEEE754
rchg_probability	Вероятность случайного отказа. Если более 0, то при моделировании HDL будут имитироваться случайные отказы элементов ПЛИС с этой вероятность.
rchg_full_mem	Определяет каким образом обеспечивается вероятность отказа памяти: для блока памяти целиком или для каждой ячейки в отдельности
rchg_show_param	Добавить служебные данные об общем количестве произошедших отказов в результаты моделирования
verilator_keys	Опциональные ключи для Verilator
coverage	Включить подсчет покрытия кода тестами
coverage_keys	Опциональные ключи конфигурации покрытия кода тестами

Процедура моделирования в MATLAB

По результатам компиляции модели vmodel создает в заданной папке 3 функции:

1) конструктор (создает структуру, через которую пользователь передает модели текущее состояние входов HDL модуля);

2) функцию моделирования заданного периода времени, при фиксированном состоянии входов модели (аналог команды # из несинтезируемого подмножества языка Verilog);

3) функцию моделирования до заданного условия (эта функция создается только в том случае, если соответствующее условие было задано в конфигурации модели).

Имена функций и папки их расположения настраиваются при помощи структуры конфигурации (табл. 1).

Используя созданные vmodel функции в скрипте на М-языке, пользователь может изменять состояние входов модели (через структуру, созданную конструктором) и проводить моделирование. Для создания тестовых воздействий и анализа результатов моделирования доступен весь широкий арсенал средств MATLAB, включающий в себя функции матричных и скалярных операций, статистического и частотного анализа, двумерного и трехмерного графического отображения.

Для иллюстрации процесса моделирования при помощи Vmodel toolbox рассмотрим скрипт на М-языке, представленный на листинге 1. Подразумевается, что к моменту выполнения данного скрипта функция vmodel уже создала в текущей папке функцию конструктора с именем «constructor», функцию моделирования заданного

периода времени (sim_step) и функцию моделирования до заданного условия (sim_tcond). При это в качестве условия остановки моделирования задано равенство единице сигнала об окончании расчета в HDL модуле.

По умолчанию время в параметрах функций моделирования указывается в тактах задающего генератора ПЛИС. Это удобно для отладки HDL-модулей с одним входом тактового генератора. В случае моделирования в режиме с несколькими тактовыми генераторами, временные параметры указываются в тактах главного генератора. Если же требуется указать время моделирования в системе СИ, то это возможно с использованием функции «t2f». В строке 6 листинга 1 представлен пример применения этой функции для установки времени моделирования в 30 мкс.

Листинг 1. Пример моделирования при помощи Vmodel_toolbox в MATLAB

```
1: uut=constructor;                    %Создаем объект uut
2: uut.rst=1;                          %Устанавливаем вход rst
                                       HDL модуля в 1
3: sim_step(uut,1);                    %Моделируем 1 такт
                                       расчета ПЛИС
4: uut.rst=0;                          %Устанавливаем вход rst
                                       HDL модуля в 0
5: uut.in=-24;                         %Устанавливаем вход in
                                       HDL модуля в -24
6: sim_step(uut,t2f(30,'us'));         %Моделируем 30 мкс
                                       работы ПЛИС
7: uut.start=1;                        %Устанавливаем вход
                                       start HDL модуля в 1
8: sim_step(uut,1);                    %Моделируем 1 такт
                                       расчета ПЛИС
9: uut.start=0;                        %Устанавливаем вход
                                       start HDL модуля в 0
10: res=sim_tcond(uut,5000);           %Моделируем работу
                                       ПЛИС до заданного
                                       условия%или в течении
                                       5000 тактов в зависимо-
                                       стей от того, % что
                                       наступит ранее. Резуль-
                                       тат помещаем в res.
11: display(res.out);                  %Отображаем значение
                                       выхода out HDL модуля
                                       в %консоль
```

Несмотря на то, что MATLAB представляет широкие средства для визуального отображения данных, в нем отсутствуют некоторые привычные для HDL разработчика способы визуализации изменения сигналов. Восполняя этот пробел, Vmodel toolbox добавляет в MATLAB функцию «digital» для отображения временных диаграмм изменения цифровых сигналов (рис. 2) и функцию «bin_stairs» для отображения побитной временная диаграммы (рис. 3).

В отличии от многих средств моделирования временные диаграммы, построенные Vmodel toolbox, позволяют отображать числовые значения сигналов не только в различных целочисленных формах, но и в виде дробных чисел с фиксированной или плавающей точкой. При этом есть поддержка систем исчисления с нестандартными размерами мантиссы и экспоненты.

Процедура моделирования в Simulink

Функция vmodel создает Simulink блок поведенческой модели HDL кода. Он имеет всего два параметра: частота работы ПЛИС и период расчета Simulink. Из соотношения этих параметров определяется то, как часто меняются значения на входах модели HDL кода и сколько тактов тактового генератора ПЛИС приходится на один период расчета среды Simulink. С точки зрения типовых задач ЦОС период расчета Simulink можно рассматривать как период обновления сигналов на выходах АЦП, подключенных к ПЛИС.

На этапе конфигурации модели можно указать, что необходимо создать не просто Simulink блок, а модель «черный ящик» для HDL Coder. В этом случае в процессе генерации кода модель будет заменена на соответствующий ей HDL код. Такой способ создания «черных ящиков» для HDL Coder обладает существенными преимуществами по сравнению с ручным созданием их из базовых блоков Simulink, так как весь процесс происходит автоматически, а на выходе разработчик получает точную модель, которая учитывает все особенности функционирования HDL-кода.

Для обеспечения совместимости среды Simulink с сигналами, имеющими нестандартную разрядность, в состав vmodel входит специальный Simulink блок Signer. Он преобразует целочисленный знаковый сигнал произвольной разрядности с выхода модели в стандартный для Simulink тип Double.

На рис. 4 представлен процесс моделирования IP-блока обработки синусно-косинусного датчика в составе модели Simulink.

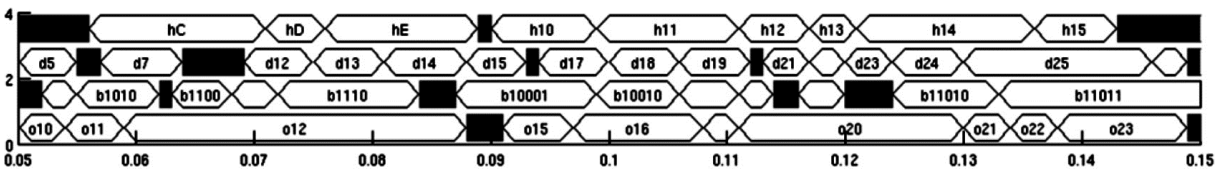

Рис. 2. Временных диаграмм изменения цифровых сигналов, построенная при помощи ПО vmodel

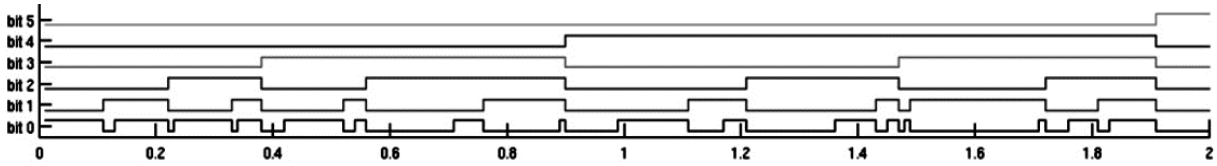

Рис. 3. Побитная временная диаграмма, построенная при помощи ПО vmodel

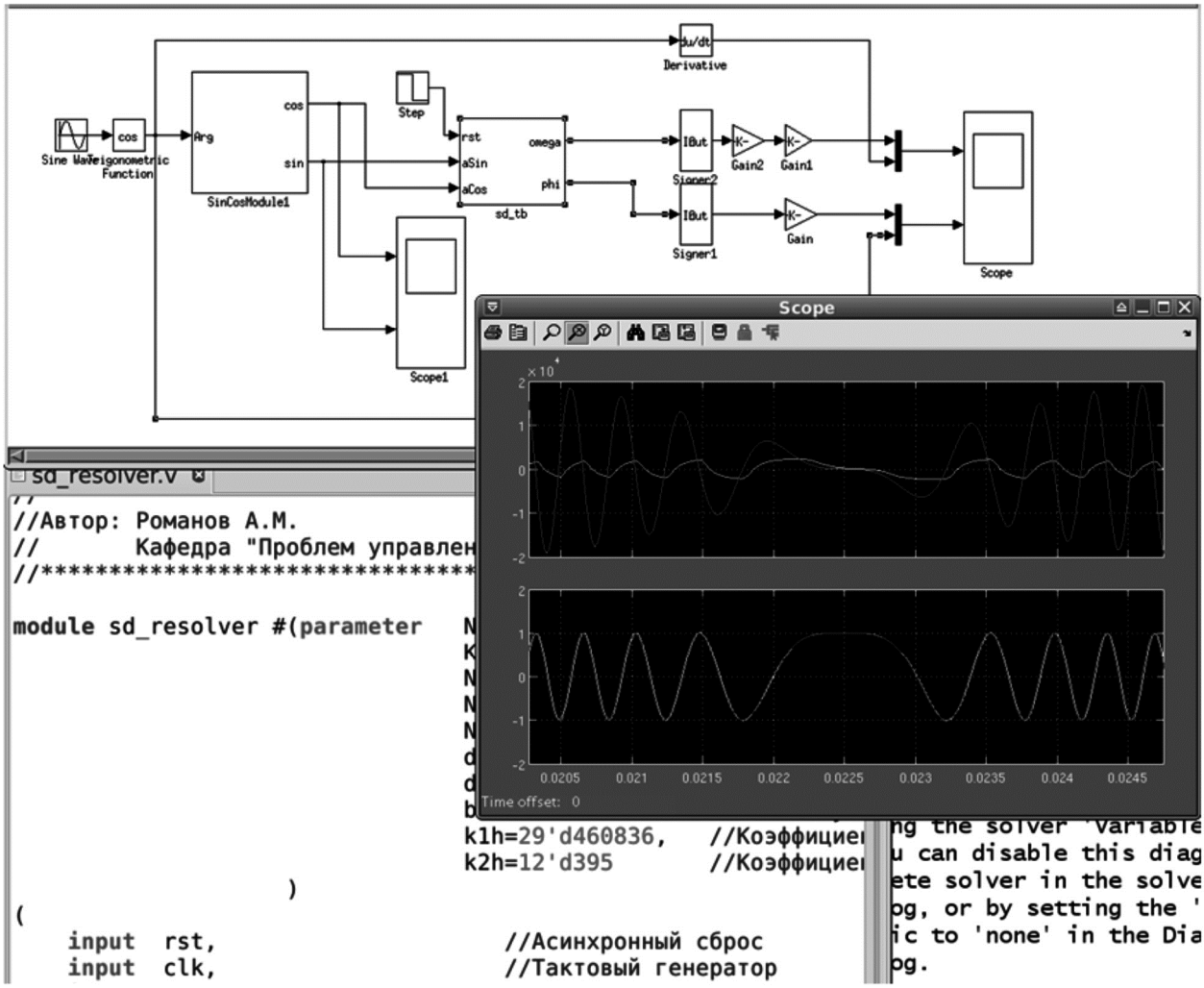

Рис. 4. Процесс моделирования IP-блока обработки синусно-косинусного датчика в составе модели Simulink

Так как на выходе Vmodel toolbox создается готовый Simulink блок, то существенных отличий по сравнению с обычным порядком моделирования в Simulink нет. Пользователю необходимо скопировать созданный блок в отдельную рабочую область Simulink и подать на него тестовые сигналы с требуемыми характеристиками. Для формирования сигналов сброса удобно использовать блок Step из стандартной библиотеки Simulink. По умолчанию выходы Simulink модели, созданной Vmodel tool-box, представлены в виде 32-битных безннаковых чисел стандарта «uint32». В то время как стандартным типом в Simulink являются числа с плавающей точкой формата Double. Для сопряжения выходов модели с другими сигналами можно использовать как стандартные блоки «Data Type Conversion» из библиотеки Simulink, так и блоки Signer, предоставляемые Vmodel toolbox. В отличии от выходных сигналов, для входных сигналов приведение типов не требуется. Вне зависимости от типа, любой поданный на модель сигнал будет округлен и подан на HDL модуль в целочисленном виде.

Передача моделей другим разработчикам

Популярный сейчас модельно-ориентированный подход проектирования [10, 11] подразумевает работу различных специалистов над общей моделью системы. Каждый из них работает над своей частью модели, используя остальные её компоненты для верификации. При таком подходе в случае использования со-симуляции специализированные HDL-симуляторы должны быть установлены на компьютерах всех участников проекта. Помимо ощутимых затрат на лицензии, это требует от специалистов, не имеющих отношения к разработке АПО ПЛИС, дополнительных компетенций. Альтернативой является использование упрощенных моделей на основе базовых блоков Simulink, которые могут быть неточными и не учитывать особенности реализации на ПЛИС.

Все модели, созданные при помощи vmodel, после компиляции могут быть использованы на любом компьютере с установленным пакетом MATLAB. Размер созданных моделей, как правило, не превышает 1 Мбайта, что позволяет без проблем распространить их между всеми участниками проекта.

В процессе создания поведенческой модели и её последующей компиляции она несколько раз оптимизируется. Таким образом, восстановить исходный код модуля по модели, созданной при помощи Vmodel toolbox, нельзя. Это позволяет беспрепятственно передавать модели IP-ядер в третьи организации, не боясь, что заложенное в них ноу-хау, будет утеряно.

Результаты внедрения

Программа Vmodel toolbox прошла государственную регистрацию и была успешно апробирована в ряде проектов, связанных с ЦОС, на основе которых можно судить об её эффективности.

Верификация матричного сопроцессора

Целью проекта было создание набора IP-ядер ПЛИС, предназначенного для ускорения матричных операций таких как: транспонирование, сложение, умножение и инверсия матриц, формирование диагональных и треугольных матриц, декомпозиция Холецкого и т.д. В результате был создан математический сопроцессор оригинальной архитектуры [12]. Для его полноценной верификации потребовалось создать большое количество тестов, включающих в себя множество матричных операций. Использование Vmodel toolbox позволило существенно упросить этот процесс, так как MATLAB уже содержит мощный аппарат матричных вычислений. В то же время функции построения временных диаграмм, входящие в состав vmodel позволили успешно отлаживать сложные конвейеры при помощи привычных для HDL-разработчиков средств, избежав применения других симуляторов.

Верификация устройств
с семисегментными индикаторами

В рамках проекта требовалось провести верификацию устройства на базе ПЛИС, одним из интерфейсов которого являлся семисегментный индикатор на 8 знаков. Применение Vmodel toolbox и встроенных функций отображения графики в MATLAB позволили вывести результаты индикации в том виде, в котором они будут отображаться пользователю конечного устройства. Такой подход существенно упростил работу инженера-тестировщика и сократил сроки верификации. Пример тестового скрипта на M-языке и отображения результатов в MATLAB представлен на рис. 5.

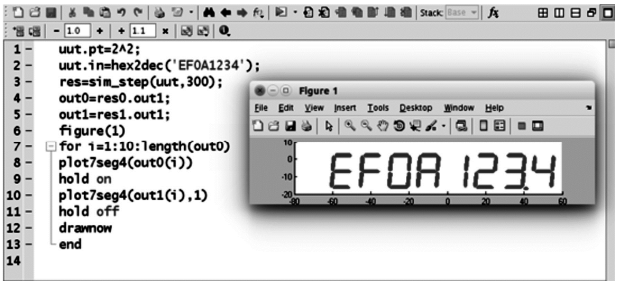

Рис. 5. Процесс моделирования IP-блока обработки синусно-косинусного датчика в составе модели Simulink

Верификация библиотеки IP-ядер
для управления двигателями

Задачей проекта была реализация библиотеки IP-ядер для систем управления двигателями различного типа. Библиотека включает в себя траекторные задатчики, регуляторы для двигателей постоянного и переменного тока, а также блоки обработки датчиков. Одним из требований проекта была совместимость IP-ядер с бюджетными ПЛИС, обладающими малой логической ёмкостью. Для выполнения этого требования большая часть ЦОС в рамках проекта была реализована при помощи «импульсных» математических операций [13]. Особенностью данного метода является представление информационных сигналов внутри блоков ЦОС в форме высокочастотной сигма-дельта модуляции. Просмотр временных диаграмм для таких сигналов не имеет смысла, а для восстановления информации из них требуется применение фильтров низких частот. Благодаря использованию Vmodel toolbox, весь анализ результатов проводился средствами MATLAB, поэтому реализация низкочастотной фильтрации не потребовала дополнительных трудозатрат. Еще одна сложность в рамках проекта заключалась в отсутствии возможности заранее сформировать тестовые воздействия: в связи с большим количеством обратных связей в системах управления двигателями входы IP-ядер на каждом такте расчета зависели от значений на выходе ядра на предыдущем такте. При этом длительность переходных процессов, которые требовалось моделировать, измерялась секундами, что соответствует сотням миллионов тактов расчета ПЛИС. Благодаря высокой скорости расчета моделей Simulink, созданных при помощи Vmodel toobox, удалось выполнить весь объем работ по верификации проекта в установленные сроки. Пример моделирования IP-ядра обработки синусно-косинусного датчика в Simulink представлен на рис. 4.

Верификация цифровой системы обработки
видео потока

При реализации на ПЛИС цифровой системы видеообработки, построенной на базе применения двумерного дискретного преобразования Фурье, стояла задача выбора оптимального значения разрядностей видеопамяти и коэффициентов преобразования. Применение Vmodel toobox позволило автоматически промоделировать тракт видеообработки при различных значения параметров HDL-модулей, определяющих требуемые разрядности. А при помощи средств визуализации MATLAB удалось отобразить содержимое видеопамяти на экране оператора без использования дополнительного программного обеспечения. В результате разработчик имеет возможность с минимальными трудозатратами оценить качество изображения в каждом случае и выбрать наиболее подходящие значения параметров модуля. Примеры результатов моделирования тракта видеообработки, полученные при различных разрядностях коэффициентов преобразования Фурье представлены на рис. 6.

Заключение

Функционал существующих средств моделирования HDL-кода не достаточен для решения задач ЦОС. Для формирования тестовых воздействий и анализа результатов моделирования разработчики используют пакеты математического моделирования, наиболее популярным из которых является MATLAB. Для задач управления, где необходимо моделировать взаимодействие блоков ЦОС на базе ПЛИС с аналоговыми объектами, единственным приемлемым решением до недавнего времени являлось использование MATLAB в режиме со-симуляции с внешним HDL-симулятором. Однако такой режим имеет низкую скорость моделирования.

а) *б)* *в)*

Рис. 1. Результаты моделирования тракта видеообработки в MATLAB при различных разрядностях коэффициентов преобразования Фурье; верхний ряд – эталонное изображение, нижний ряд — результат обработки при помощи IP-ядра; а) 2x16 бит на точку б) 2x20 бит на точку в) 2x28 бит на точку

Предложенный в данной работе инструмент Vmodel toolbox позволяет проводить отладку HDL-кода без использования внешних HDL-симуляторов, только при помощи интерфейса MATLAB. Предварительная компиляция поведенческих моделей, которая используется в Vmodel toolbox, обеспечивает не только высокую скорость моделирования, но и возможность их использования на компьютерах без специализированного программного обеспечения для работы с HDL-кодом. Благодаря этому существует возможность передать точную поведенческую модель IP-ядра третьим лицам без раскрытия исходного кода.

Использование средств MATLAB для формирования тестовых воздействий, анализа и визуализации результатов моделирования позволяет поднять верификацию блоков ЦОС на новый уровень. Тестовое окружение, созданное на этапе прототипирования алгоритма ЦОС, может быть без изменений использовано и для отладки HDL-кода, а для визуализации результатов моделирования больше не требуется выгружать данные в стороннее программное обеспечение.

Совместимость Vmodel toolbox с HDL Coder позволяет повысить эффективность работы с этим инструментом, за счет полной автоматизации создания функциональных моделей существующих IP-ядер. Опыт успешного применения программного обеспечения Vmodel toolbox в реальных проектах показал, что оно не только удовлетворяет всем требованиям, предъявляемым к средствам верификации АПО ПЛИС в задачах ЦОС, но и позволяет обойтись без использования каких-либо других симуляторов при отладке HDL-кода.

Литература

1. Попов А.Ю. Проектирование цифровых устройств с использованием ПЛИС: Учеб. пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 80 с.

2. Sandre-Hernandez, O.; Rangel-Magdaleno, J.J.; Morales-Caporal, R., «Simulink-HDL cosimulation of direct torque control of a PM synchronous machine based FPGA,» Electrical Engineering, Computing Science and Automatic Control (CCE), 2014 11th International Conference on, vol., no., pp.1, 6, Sept. 29 2014-Oct. 3 2014

3. Besbes, M.; Said, S.H.; M'Sahli, F., «FPGA implementation of high gain observer for induction machine using Simulink HDL

coder,» in Control, Engineering & Information Technology (CEIT), 2015 3rd International Conference on , vol., no., pp.1-6, 25-27 May 2015

4. Свидетельство о государственной регистрации программы для ЭВМ №2015612744 (Vmodel toolbox)

5. Wilson S. «Introduction to Verilator» [Электронный ресурс] = Intro-Verilator-Veripool – Режим доступа: http://www.veripool.org/wiki/verilator, свободный. – Загл. с экрана.

6. Wilson S. «Verilog Simulator Benchmarks» [Электронный ресурс] = Verilog Simulator Benchmarks-Veripool – Режим доступа: http://www.veripool.org/wiki/veripool/Verilog_Simulator_ Benchmarks, свободный. – Загл. с экрана.

7. Ahmad, T.B.; Ciesielski, M., «Parallel Multi-core Verilog HDL Simulation Using Domain Partitioning,» in VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on, vol., no., pp.619-624, 9-11 July 2014

8. T.A. Alspaugh «GNU Lesser General Public License, version 3» [Электронный ресурс] = LGPLv3 numbered – Режим доступа: http://www.thomasalspaugh.org/pub/osl-sps/lgpl3.0.html, свободный. – Загл. с экрана.

9. Романов A.M. Vmodel toolbox repository [Электронный ресурс] = amromanov/vmodel – Режим доступа: https://github.com/amromanov/vmodel, свободный. – Загл. с экрана.

10. Sudhir Sharma, Wang Chen, «Using Model-Based Design to Accelerate FPGA Development for Automotive Applications», in 2009 SAE World Congress, January 1-15, 2009.

11. Kelemenová, Tatiana, et al. «Model Based Design and HIL Simulations» American Journal of Mechanical Engineering 1.7 (2013): 276-281.

12. A. Romanov, B.Slaschov FPGA-based Kalman filtering for motor control //NSCE2014 Transactions, CRC Press, 2014 г.

13. А.М. Романов Анализ и синтез элементов устройств управления мехатронно-модульными системами на базе ПЛИС с использованием сигма-дельта модуляции // Естественные и технические науки. – М.: «Компания Спутник+», 2013. – № 6.