Методика аппаратно-программного моделирования и тестирования проектируемых систем

Владимир Вычужанин (г. Одесса, Украина)

В статье описана методика применения программного обеспечения для моделирования и тестирования в реальном времени проектируемых аппаратных средств цифровой обработки сигналов на основе ПЛИС.

Введение

В настоящее время проектировщики используют различные методы разработки систем цифровой обработки сигналов (ЦОС), позволяющие настраивать и тестировать подобные системы в реальном времени. Проектирование систем ЦОС преимущественно осуществляется на основе двух подходов.

При первом подходе функциональность и производительность системы ЦОС исследуют и настраивают с помощью программного обеспечения (ПО) инструмента моделирования. Уровень доверия к полученным результатам определяется соответствием модели системы заданному проекту. Программное обеспечение моделирования обеспечивает гибкость проверки системы, позволяя исследовать различные сценарии её поведения. Однако модель, основанная на реальной системе, при использовании только ПО моделирования обладает естественными ограничениями:

- время моделирования системы может быть длительным при исследовании сложных систем. Например, в электромеханических системах постоянная времени может измеряться в секундах, в то время как время вычисления параметров системы измеряется в микросекундах;
- ошибки в вычислениях из-за неполного описания системы моделью. Кроме того, некоторые элементы системы могут содержать нелинейные компоненты и подвергаться воздействиям окружающей среды, которые сложно учесть;
- трудности, связанные с проверкой надёжности проекта.

Второй подход заключается в разработке проекта на целевой платформе для проверки работоспособности и оценки производительности оборудования системы ЦОС в реальных условиях. При реализации необходимо запустить для исследования одновременно как само устройство, так и его модель, получить набор данных и сравнить полученные результаты. Этот подход, так же как и первый, имеет свои недостатки:

- обычно испытания системы производятся в предопределённых сценариях эксплуатации;
- на начальных этапах проектирования не всегда доступна конечная реализация оборудования системы;
- стоимость тестирования зависит от выбора оборудования;
- при неопределённости в выборе системных компонентов необходимо уделять особое внимание безопасности.

Подход HIL

Преимущества обоих подходов сочетает метод аппаратно-программного моделирования Hardware in the Loop (HIL, включение оборудования в петлю управления). Методология HIL [1–5] обеспечивает промежуточный уровень контроля разрабатываемых систем – между программным обеспечением моделирования и тестированием оборудования проекта. При использовании HIL-подхода проект разворачивается аппаратно и работает в режиме реального времени. При запуске модели в Simulink происходит её взаимодействие непосредственно с аппаратной реализацией проекта в ПЛИС, как единого целого в процессе моделирования. При этом рекурсивные алгоритмы, использующие обратные связи, являются простыми и точными. Например, при рекурсивной адаптивной фильтрации контуры управления и БИХ-фильтры обычно имеют обратные связи, как часть проекта. Эти устройства подвержены ошибкам, которые могут накапливаться и усиливаться при итерациях, что может привести к неустойчивости проектируемой системы. HIL-подход с программным моделированием проекта позволяет обнаружить ошибки на ранних стадиях разработки системы.

С использованием HIL проектировщик разрабатывает систему ЦОС и запускает симуляции моделей, описывающих её аппаратные и программные составляющие, а окружающие систему компоненты (генераторы, датчики, измерительные устройства и т.д.) моделируются в программной среде. Рабочие данные могут быть загружены на ПК и проанализированы с использованием различных средств. Такая методика позволяет сочетать гибкость программного обеспечения с реальной точностью аппаратуры и ускорить выполнение разработки.

Традиционно проектирование системы ЦОС состоит из следующих этапов (это относится ко всем встроенным системам управления на базе микроконтроллеров и ПЛИС – *прим. ред.*):

- формализация технического задания (ТЗ) с разработкой структуры системы;
- построение математической модели (например, в среде MATLAB/Simulink) и моделирование системы;
- перевод математической модели проектируемой системы в текстовое описание с использованием, например, языка HDL (Verilog, VHDL);
- разработка тестов для проверки соответствия математической модели созданного HDL-описания проектируемой системы с последующим моделированием;
- реализация проекта на базе, например, ПЛИС с проверкой соответствия требуемых аппаратных затрат, быстродействия и т.д. требованиям, установленным ТЗ;
- отладка системы.

В основе методики аппаратно-программного моделирования проектируемых систем ЦОС лежит интеграция перечисленных этапов в единый итерационный цикл проектирования, с включением автоматизации процес-

са передачи данных (HDL-кода) между этапами математического описания проекта и от одной модели к другой.

Для перехода от математической модели к аппаратной реализации алгоритма функционирования проектируемой системы можно использовать программы Simulink HDL Coder от MathWorks из пакета MATLAB, Altera DSP Builder или Xilinx System Generator. MATLAB/Simulink является основной средой для моделирования и разработки систем ЦОС. Она позволяет осуществлять иерархическое моделирование большого, расширяемого набора блоков системы и снабжена открытым интерфейсом разработчика (API). В пакете MATLAB/Simulink для описания устройств могут использоваться встроенные языки M, C или параметризируемые модули из библиотеки Simulink. При переводе математической модели проектируемой системы в текстовое описание используется язык HDL.

Применение продуктов MathWorks для генерации встраиваемого кода позволяет:

- генерировать C/C++ код;
- интегрировать полученный код с кодом, написанным вручную;
- профилировать и осуществлять верификацию встраиваемого проекта на ПЛИС.

Также имеется возможность транслировать функционал модели системы ЦОС в представление Verilog и VHDL-код для ПЛИС. Используя пакеты Altera DSP Builder или Xilinx System Generator необходимо заменить стандартные блоки пакета Simulink на аналогичные из состава пакетов аппаратной разработки ПЛИС. В результате проектируемое устройство реализуется на целевой СБИС и выполняется совместное моделирование (алгоритм реализован на СБИС, а входные и выходные данные передаются, получаются и анализируются с помощью ПК в рамках пакета MATLAB/Simulink). Для проведения тестирования используется стандартный подход, реализованный в пакетах DSP Builder и System Generator – HIL. Полученные результаты должны совпадать с результатами, полученными при программном моделировании системы.

К преимуществам данной методики следует отнести автоматизацию процесса перехода от математической модели проектируемого устройства к его аппаратной реализации на основе генерации встраиваемого кода, что позволяет повысить производительность труда проектировщика и улучшить качество разработки.

Чтобы реализовать разработанные в MATLAB/Simulink алгоритмы на ПЛИС Altera, для создания, моделирования и верификации алгоритма в среде Simulink следует использовать DSP Builder от Altera. В библиотеке предусмотрены блоки для оценки выполнения алгоритма на устройствах Altera, генерации оптимизированного по времени кода HDL и верификации аппаратной реализации с применением эталонных моделей Simulink.

Блоки из библиотеки DSP Builder предоставляют возможность создать аппаратную реализацию моделируемой в Simulink системы. Для проверки соответствия математической модели созданного HDL-описания проектируемого устройства используется пакет Quartus II, а также специализированные платы с целевой ПЛИС и набором необходимых периферийных компонентов. Чтобы оценить реализацию алгоритма на ПЛИС в среде Simulink-Avalon следует использовать набор блоков Altera, выполненных с точностью до бита и такта, где реализованы арифметические и логические функции, интерфейсы памяти и схем распределения памяти и потоков. В проект может быть внедрён код HDL и модели из Altera MegaCore IP.

Оптимизация разработанной системы цифровой обработки сигналов под ПЛИС – длительный процесс, требующий применения технологических приёмов работы с кодом HDL. Набор блоков DSP Builder позволяет реализовывать оптимизированные под ПЛИС проекты за считанные минуты, не обращаясь непосредственно к коду HDL. Разработчик определяет ограничения высокого уровня, например, частоту синхронизации и число каналов модели Simulink. Далее при помощи DSP Builder генерируется RTL-схема с конвейерной организацией для оптимизации и использования выбранной ПЛИС. Расширенный набор блоков использует временное мультиплексирование для оптимизации используемой логики и позволяет автоматически внедрять конвейеры и регистры для учёта заданных ограничений. В результате полученный код для ПЛИС будет иметь характеристики, схожие с оптимизированным вручную кодом HDL.

Рекомендуемые этапы разработки включают синтез кода, компиляцию в рамках проекта Quartus II, модели-

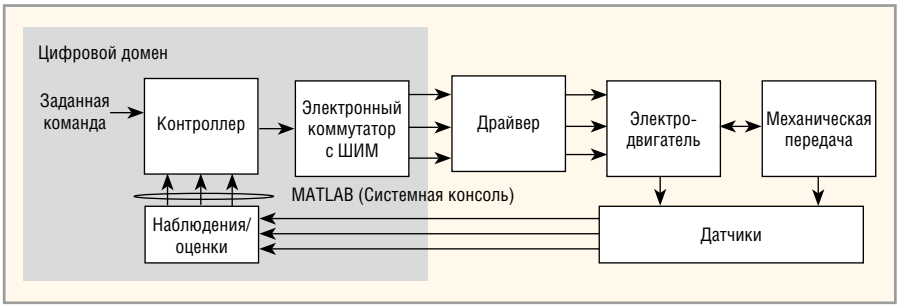

Рис. 1. HIL с MATLAB API проекта системы ЦОС привода электродвигателя

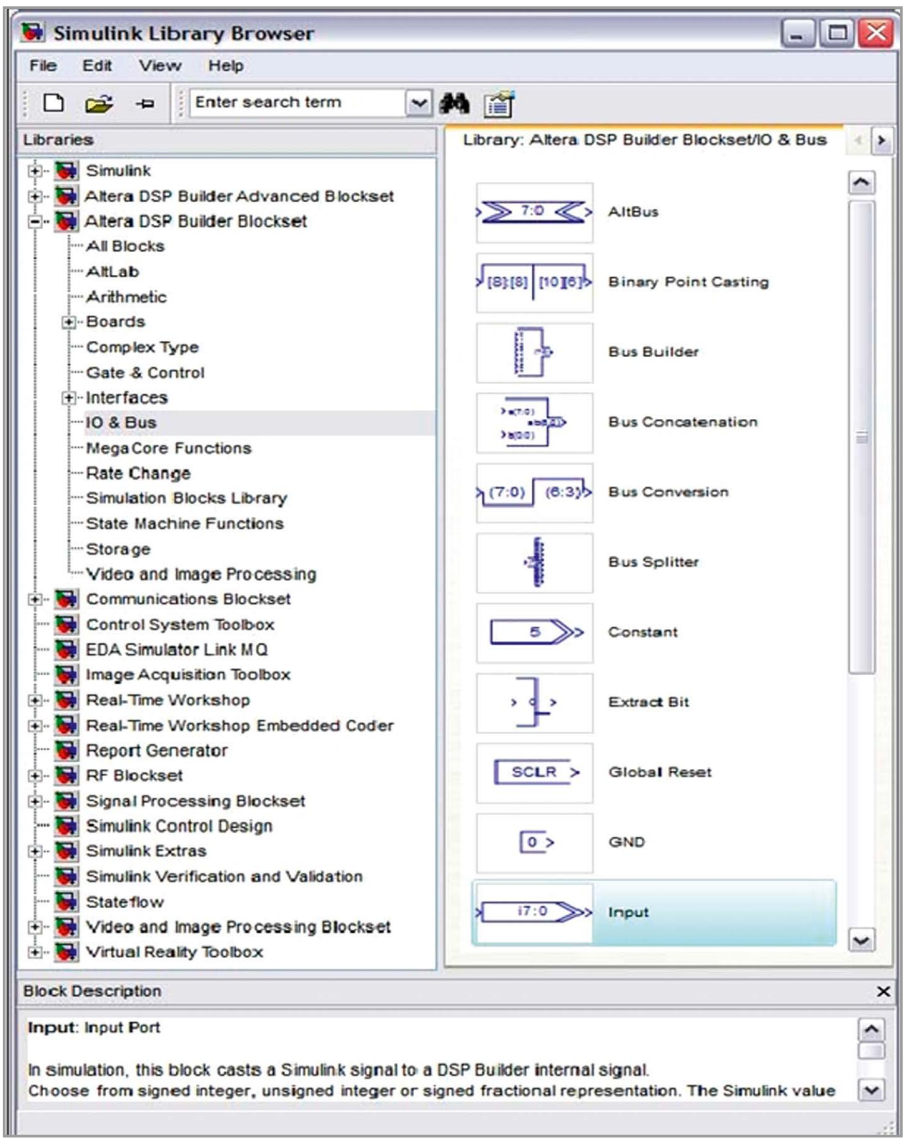

Рис. 2. Библиотека Altera DSP Builder Blockset в окне Simulink Library

Листинг 1. Пример скрипта MATLAB API

```
 SystemConsole.refreshMasters;  %Investigate available targets
M = SystemConsole.openMaster(1); %Creates connection with FPGA target
%%%%%%%% User Application %%%%%%%%%%%%
. . . .
M.write('uint321/write_address/data); %Send data to FPGA target
. . . .
data = M.read('uint321,read_address,size);  %Read data from FPGA
target
%%%%%%%%%%%%%%%%%%%%
M.close; %Terminates connection to FPGA target
```

рование с использованием автоматически сгенерированного экспериментального стенда HDL и загрузку программы на одну из поддерживаемых макетных плат. Для ускорения верификации с использованием средств визуализации и анализа Simulink следует компилировать проект по частям и выполнять параллельное моделирование на ПЛИС и эталонных моделях Simulink.

При использовании методик полунатурного и быстрого прототипирования, связанного с хост-компьютером, программное обеспечение ПК моделирует поведение тестируемого устройства, а внешние компоненты являются реальными устройствами. Это позволяет получить точные настройки проектируемой системы на основе моделей. Для инициирования и поддержания связи между двумя платформами коммуникационной инфраструктуры можно использовать ПЛИС на основе системной консоли (System Console), представляющей собой гибкую систему отладки проекта и позволяющей осуществлять операции записи/чтения в проектируемой системе. Для размещения разработки необходимо увязать проект с ПЛИС, обладающей также интерфейсом прикладного программирования (API) в среде MathWorks MATLAB/Simulink для обмена данными.

Пример использования MATLAB API

Интерфейс MATLAB API можно использовать в задачах проектирования при оценке проекта, а также для настройки ПЛИС. Его возможности наиболее полно проявляются в HIL-подходе, что позволяет создавать на ПК сложные варианты переборов данных в режиме реального времени и загружать их на целевую платформу ПЛИС, а также контролировать, отлаживать, визуализировать и верифицировать проект на ПЛИС в среде MATLAB.

Протокол MATLAB API представляет собой простой набор команд [6], позволяющий выполнять в реальном времени взаимодействие проекта на целевой ПЛИС с хост-компьютером с помощью отображённых в памяти операций, инициированных MATLAB. Операции захватывает специальный компонент с распределённой памятью (Avalon-MM). После обработки данных для ПЛИС результат загружается на хост-компьютер с помощью MATLAB API. При этом можно исполь-

зовать сложные методы анализа данных с последующим графическим отображением, доступным в MATLAB. Подход HIL с MATLAB API может быть использован для сравнения программной модели, теоретических результатов и предыдущих реализаций проекта. Пример скрипта MATLAB API приведён в листинге 1.

Гибкость MATLAB API позволяет применять его в Simulink при моделировании проекта, например, привода бесщёточного электродвигателя (см. рис. 1). Для связи проектируемого привода с ПЛИС в Simulink реализуются коммуникационные порты и применяются S-функциональные блоки из библиотеки Altera DSP Builder Blockset в окне Simulink Library (см. рис. 2). Эти блоки позволяют описать функциональные устройства проекта с использованием языка программирования высокого уровня (скрипты MATLAB, язык C). При моделировании в Simulink работоспособность блоков оценивается на основе цикла моделирования. MATLAB API использует фактическое описание S-функциональных блоков для записи/считывания данных

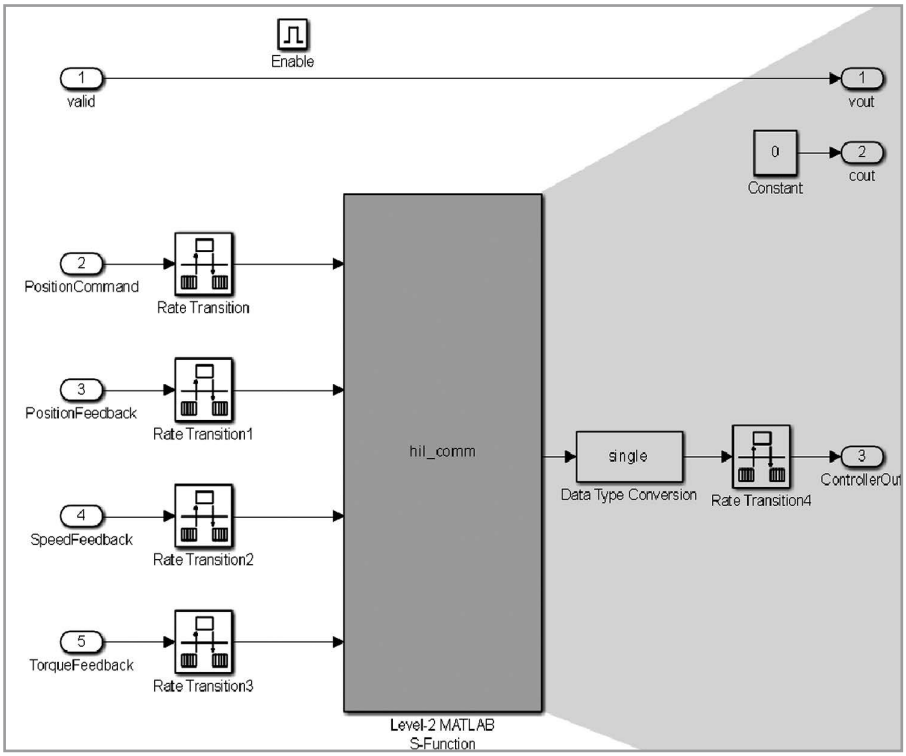

Рис. 3. Связь HIL с реализацией проекта с MATLAB Simulink

в ПЛИС. В листинге 2 приведён программный файл записи выборки данных в ПЛИС, где результаты обработки в ПЛИС считываются с помощью MATLAB API и становятся доступными блокам Simulink (см. рис. 3).

Листинг 2. Программный файл записи выборки данных в ПЛИС

```
1 function hil_comm(block)
2 setup(block);
3 %endfunction
4 function setup(block)
5 block.NumDialogPrms = 0;
6 %% Register number of input and output ports
7 block.NumlnputPorts = 4;
8 block.NumOutputPorts = 1;
9 %% Setup functional port properties
10 block.InputPort(1).Dimensions     = 1;
11 block.InputPort(1).DirectFeedthrough = false;
12 block.InputPort(1).DatatypelD     = 1;
13 block.InputPort(2).Dimensions     = 1;
14 block.InputPort(2).DirectFeedthrough = false;
15 block.InputPort(2).DatatypelD     =1;
16 block.InputPort(3).Dimensions     =1;
17 block.InputPort(3).DirectFeedthrough = false;
18 block.InputPort(3).DatatypelD     = 1;
19 block.InputPort(4).Dimensions     = 1;
20 block.InputPort(4).DirectFeedthrough = false;
21 block.InputPort(4).DatatypelD     = 1;
22 block.OutputPort(1).Dimensions    = 1;
23 block.OutputPort(1).DatatypelD    = 1; %single
24 %% Set block sample time to inherited
25 global SimulinkSampleTime
26 block.SampleTimes = [SimulinkSampleTime 0];
27 %% Register methods
28 block.RegBlockMethod('Outputs',   @Output);
29 block. RegBlockMethod (' Terminate ',    @Terminate);
30 block.RegBlockHethod(' Enable',      @DoZnable);
31 %endfunction
32 function DoZnable(~)
33 SystemConsole.refreshHasters;
34 global M
35 global SysConlndexPath
36 M = SystemConsole.openMaster(SysConlndexPath);%may be need to
modify to ajust to hw
37
38 function Output(block)
39 global M
40 InputData = [];
41 for i = 1:4
42     InputData = [InputData block.InputPort(i).Data];
43 end
44 %assumes that all inputs are single precision
45 InputData = uint32(hex2dec(num2hex(InputData)));
46 H.write ('uint32',uint32(0),InputData);
47     %toggle valid
48 M.write{'uint32',uint32(4*4), 1);
49 M.write('uint32',uint32(4*4), 0);
50 block.OutputPort(1).Data = typecast(uint32(M.
read('uint32',uint32(5*4),1)), 'single');
51 %endfunction
52
53 function Terminate (~)
54 global M
55 if (ismethod(M,'close'))
56     M.close;
57 end
```

Проект системы ЦОС для привода электродвигателя реализуется в графической модели с помощью MATLAB/Simulink с использованием потактового моделирования. Применение подобной методологии для проекта на ПЛИС позволяет создать схему управления приводом электродвигателя путём моделирования в Simulink по генерируемому коду (см. рис. 4), а также обеспечить визуализацию результатов моделирования (см. рис. 5).

Аппаратная реализация модели привода электродвигателя (см. рис. 4) осуществлена с помощью технологии синтеза DSP Builder Advanced Blockset в Simulink для ПЛИС с использованием HIL [7]; DSP Builder, в данном случае, – это переводчик из графики Simulink в используемый HDL-код. Данная технология позволяет автоматически генерировать оптимизированный код регистровых передач (RTL), соответствующий уровню Simulink для описания проекта. Для проверки DSP Builder используется инструмент HIL. Собирается проект исследуемой системы, а затем создаётся «накрывающая» модель, где проектируемая система является HIL-подсистемой, отделённой от модельного пространства входами и выходами (Input и Output, соответственно).

Непосредственно из моделей может быть автоматически сгенерирован настраиваемый код VHDL или Verilog, которые используются на любой целевой вычислительной аппаратуре и синтезируются для реализации на ПЛИС. Код можно быстро модифицировать, обновляя модель и повторно выполняя автоматическую генерацию. Далее при помощи сторонних средств выполняется синтез, размещение, трассировка и загрузка окончательной версии проекта в ПЛИС.

Для верификации реализованного на ПЛИС алгоритма можно использовать модель Simulink при параллельном моделировании. Программа EDA Simulator Link позволяет исполнять реализацию на имитаторах HDL, например на ModelSim, применяемом в качестве эталонной модели. Сравнением результатов выполнения реализаций алгоритма определяется разница между полученным и желаемым поведением системы. Кроме этого, EDA Simulator Link позволяет проводить регрессионное тестирование и интерактивную отладку как в Simulink, так и в имитаторе HDL. Такой подход избавляет разработчика

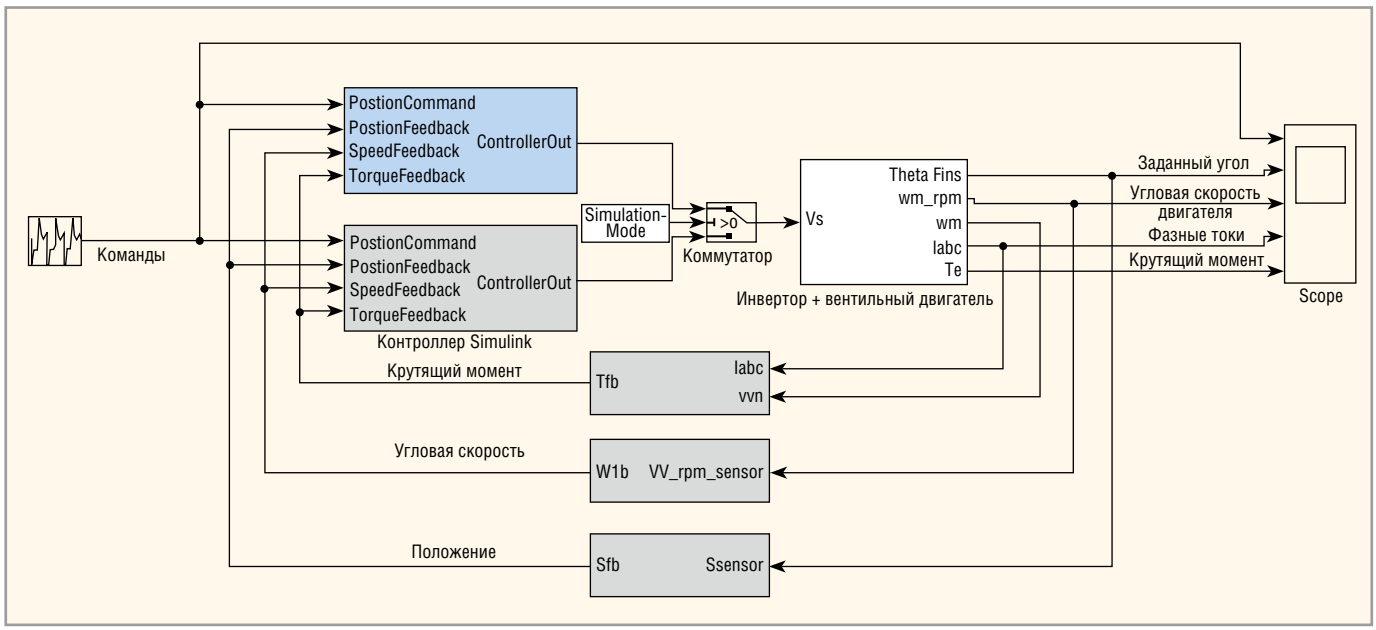

Рис. 4. Реализация системы ЦОС привода электродвигателя в Simulink с DSP Builder Advanced Blockset

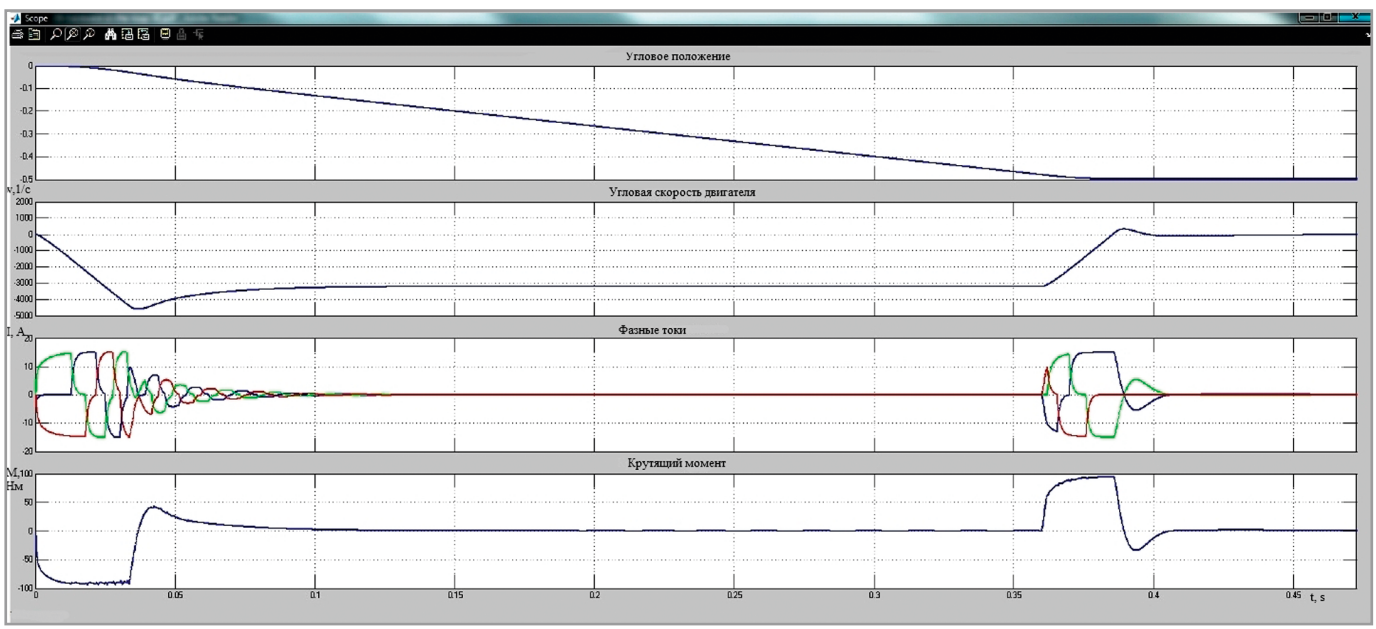

Рис. 5. Визуализация HIL-связи устройств модели привода электродвигателя в MATLAB/Simulink

от ручного переноса тестовых векторов из одной среды в другую и даёт возможность обнаружить ошибки на ранних стадиях проекта.

Для параллельного моделирования и верификации элементы моделей Simulink, созданные с применением System Generator для DSP, могут быть загружены на макетную плату и ассоциированы с соответствующим блоком в модели. Программа System Generator для DSP поддерживает интерфейсы Ethernet и JTAG для соединения аппаратной платформы и Simulink. Как уже было отмечено, одними из существенных преимуществ использования среды MATLAB/Simulink являются анализ данных и возможности визуализации. Реализация HIL с ПЛИС Altera и MATLAB API позволяет осуществить анализ данных в реальном времени, а также ускорить время моделирования. Следует отметить, что с увеличением сложности проекта системы ЦОС, преимущества такой реализации становятся ещё более очевидными.

Литература

1. *Halvorsen Hans-Petter.* Introduction to Hardware-in-Loop Simulation. Telemark University College. 2012.
2. *Hwang T., Robl J., Park K., Hwang J., Lee K. H., Lee K., Lee S.-J., Kim Y.-J.* Development of HIL Systems for Active Brake Control Systems. SICE-ICASE International Joint Conference. 2006.
3. *Raman S., Sivashankar N., Milam W., Stuart W., Nabi S.* Design and Implementation of HIL Simulators for Powertrain Control System Software Development. Proceedings of the American Control Conference.1999.
4. *Cebi A., Guvenc L., Demirci M., Karadeniz C., Kanar K., Guraslan E.* A Low Cost, Portable Engine Electronic Control Unit Hardware-in-the-Loop Test System. Proceedings of the IEEE International Symposium on Industrial Electronics. 2005.
5. *Du J., Wang Y., Yang C., Wang H.* Hardware-in-the-Loop Simulation Approach to Testing Controller of Sequential Turbocharging System. Proceedings of the IEEE International Conference on Automation and Logistics. 2007.
6. www.altera.com/literature/hb/qts/qts_qii53028.pdf.
7. www.altera.com/literature/hb/dspb/hb_dspb_std.pdf.