Министерство образования и науки Российской Федерации
Государственное образовательное учреждение высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

УТВЕРЖДАЮ

Проректор по учебной работе
_____ Л.А. Боков
«____» _____ 2011 г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
К ПРАКТИЧЕСКИМ ЗАНЯТИЯМ

по дисциплине

«Основы проектирования систем на кристалле. Часть II»

Составлена кафедрой «Управление инновациями»

Для студентов, обучающихся
по направлению подготовки 220600 «Инноватика»
Магистерская программа «Мультимедийные многопроцессорные системы на кристалле»

Форма обучения очная

Составитель
Доцент _____ О.Г. Пономарев

«___» _____ 2011 г.

Томск 2011

Содержание

Разработка и моделирование цифровых устройств средствами MatLab/Simulink

Введение в Simulink. Основные понятия

Simulink – это программная среда, предназначенная для моделирования и анализа динамических систем, т.е. систем, состояние и выходные сигналы которых меняются с течением времени. Моделирование динамических систем в Simulink происходит в два этапа. На первом – пользователь, используя готовые блоки, создает в редакторе пакета Simulink модельную диаграмму системы. В этой диаграмме графически представляются математические зависимости от времени между значениями входных и выходных параметров системы и ее состоянием. На втором этапе пользователь запускает моделирование системы, представленной диаграммой, указывая при этом временной интервал работы.

Диаграмма модели в Simulink состоит из блоков (*block*), соединенных сигналами (*signal*). Каждый блок, сам по себе, представляет собой некоторую элементарную динамическую систему. Каждый блок имеет порты (*port*) для подключения входных и выходных сигналов. Значения выходных сигналов блока определяются текущими и, возможно, предыдущими значениями некоторой переменной, меняющейся во времени, которая в Simulink определяет состояние блока (*state*). Математическая зависимость между входными значениями, состоянием блока и выходными значениями определяется обыкновенным дифференциальным уравнением. В процессе моделирования поведения системы заданный пользователем временной интервал разбивается на подынтервалы (*time step*). На каждом подынтервале находится численное решение уравнения для каждого блока, что позволяет определить значения выходных сигналов блока на каждом следующем временном шаге. Для численного решения уравнений в Simulink доступно несколько алгоритмов интегрирования (*solver*). Точность получаемого решения, конечно, сильно зависит от величины временного шага. Некоторые из алгоритмов интегрирования в Simulink адаптируют величину временных подынтервалов в зависимости от скорости изменения состояния системы (*variable step solver*). В других интеграторах используется постоянный размер временных интервалов, который задается пользователем при настройке модели (*fixed step solver*).

Simulink предназначен не только для моделирования динамических систем, состояние которых постоянно (непрерывно) меняется во времени. В этом пакете предусмотрены богатые средства, позволяющие моделировать системы дискретного времени. Это, прежде всего, готовые блоки дискретного времени и два типа вычислителей (*discret solver*), рассчитывающих состояние дискретных блоков на каждом временном шаге. При этом результат вычислений (состояние блока и значения его выходных сигналов) определяются состоянием блока на предыдущем шаге и значениями входных сигналов. Один из этих вычислителей определяет состояние каждого из блоков модели через равные промежутки времени (*fixed step discret solver*). Второй – выбирает каждый временной шаг, исходя из моментов, в которые состояние системы действительно меняется (*variable step discret solver*).

Большинство стандартных блоков, поставляемых с Simulink, параметризованы. Так, например, значение константы блока **Constant** является параметром этого блока. Значения параметров параметризованных блоков задаются в специальном диалоговом окне, открывающемся при двойном щелчке манипулятора «мышь» по каждому такому блоку. Значения параметров таких блоков, задаваемые выражениями языка MatLab, могут меняться пользователем при разработке модели. Например, в одной модели может присутствовать несколько блоков **Constant**, с различными значениями констант.

В документации к Simulink выделено понятие *tunable parameter*. Значения таких параметров могут меняться пользователем непосредственно в процессе симуляции системы, т.е. изменение таких параметров не требует перекомпиляции модели.

Для каждого блока, используемого в Simulink-модели, должен быть определен временной шаг (дискрет). Именно для этого шага выполняется вычисление состояния каждого блока. Большинство блоков позволяют задавать временной дискрет как параметр. Для блоков непрерывного времени (состояние которых меняется во времени непрерывно) величина дискрета считается бесконечно малой величиной. Однако, в Simulink есть ряд блоков, для которых нет возможности задать временной шаг в явном виде. У таких блоков дискрет во времени определяется по моментам изменения входных сигналов. При этом, если хотя бы один из входных сигналов имеет непрерывное время, то данный блок считается блоком непрерывного времени, а его временной дискрет устанавливается в бесконечно малую величину. В противном случае, блок считается блоком дискретного времени. Если временные дискреты всех входных сигналов кратны наименьшему из них, то временной шаг блока устанавливается равным этому наименьшему шагу. Если это не так, то временной шаг блока устанавливается равным *fundamental sample time*, то есть равным временному интервалу, являющемуся наибольшим целым общим делителем дискретов входных сигналов.

Библиотека готовых блоков, поставляемая с Simulink, не исчерпывает конечно все варианты «строительных блоков», которые могут понадобиться при разработке какой-либо модели. В связи с этим в Simulink предусмотрена возможность создания блоков пользователем. Такие блоки называют *custom blocks*. Доступно несколько вариантов создания таких блоков. Во-первых, *custom block* можно создать графически, соединяя проводниками-сигналами несколько стандартных блоков Simulink и поместив получившуюся диаграмму в специальный блок из библиотеки Simulink, называемый *subsystem*. Созданный таким образом *custom*-блок можно параметризовать, добавив к нему диалоговое окно для задания значений параметров, используя для этого *block mask*. Вторая возможность создания *custom*-блока – описать его функциональность программно. Здесь есть два варианта. Можно использовать *Embedded MatLab block*, функциональность которого описывается на урезанной версии языка программирования MatLab в специальном редакторе. Второй вариант предоставляет существенно более богатые возможности и заключается в написании системной функции блока, называемой в документации *S-function*.

Полезность блоков *subsystem* не исчерпывается только использованием их при графическом создании *custom*-блоков. Подсистемы (*subsystems*) позволяют упорядочить и структурировать графическое представление диаграммы модели. Для этого набор блоков диаграммы, объединенных логически, объединяется в подсистему, представляемую в диаграмме единым блоком. Двойной щелчок «мыши» по такому блоку открывает его содержимое в отдельном окне графического редактора.

С подсистемами в Simulink связано еще несколько понятий. Во-первых, в Simulink предусмотрена возможность создания условных подсистем, блоки которых активизируются только при выполнении некоторого условия (*conditionaly executed subsystem*). Во-вторых, различают *virtual* и *atomic* подсистемы. Виртуальные (*virtual*) подсистемы являются просто способом организации диаграммы модели. Их наличие никак не сказывается на выполнении модели в процессе симуляции. Если подсистема является *atomic* подсистемой, то весь набор уравнений, который она описывает, интегрируется как единое целое. Таким образом, наличие *atomic* подсистемы в модели может поменять порядок выполнения вычислений при симуляции модели. По умолчанию все не условные подсистемы являются виртуальными (*virtual*). Условные подсистемы являются *atomic* подсистемами. У пользователя есть возможность объявить любую подсистему как *atomic* подсистему.

Помимо блоков любая модель в Simulink содержит сигналы, соединяющие блоки в диаграмму. Сигнал в Simulink – это меняющаяся во времени величина, значения которой заданы в каждый момент времени при симуляции модели. При разработке модели пользователь может задавать множество атрибутов сигналов: имя сигнала, тип данных

(например, uint8, int16, double и т.д.), являются значения этого сигнала вещественными или комплексными (*numeric type*), размерность сигнала (скаляр, вектор, массив, многомерный массив). При создании сигналов на диаграмме модели они отображаются в виде стрелок. Направление стрелки указывает какой из блоков, соединенных сигналом является источником значений сигнала, а какой – приемником. Источник в процессе выполнения вычислений задает (пишет в сигнал) значения сигнала в каждый момент времени. Приемный блок читает значения сигнала, используя их в вычислениях.

Итак, блоки в Simulink представляют обыкновенные дифференциальные уравнения. Решение уравнений при моделировании системы производится при вызове соответсвующих функций-методов каждого блока (*block methods*). Вызов этих методов производится в процессе, называемом *simulation loop* (цикл симуляции). Каждая итерация этого цикла представляет, таким образом, изменение состояния моделируемой системы в соответствующий момент времени.

Можно выделить три наиболее общих типа методов блоков:

- *outputs* – метод вычисляет значения выходных сигналов блока на основе значений входных сигналов в данный момент времени и состояния блока в предыдущий момент времени;
- *update* – вычисляет состояние блока дискретного времени на основе значений входных сигналов в данный момент времени и состояния блока в предыдущий момент времени;
- *derivatives* – вычисляет значение производной состояния блока (для блоков непрерывного времени) на основе значений входных сигналов в данный момент времени и состояния блока в предыдущий момент времени.

Кроме методов каждого блока в Simulink определены методы модели в целом (*model methods*). Эти методы вызываются в процессе моделирования для определения свойств и значений выходных сигналов модели в целом. Как правило, работа *model methods* заключается в вызове соответствующих методов всех блоков модели (*block methods*).

Процесс симуляции в Simulink происходит в несколько этапов. Первый из них – компиляция модели. На этом этапе производится расчет всех значений параметров блоков, заданных выражениями на языке программирования MatLab; определение атрибутов сигналов не заданных пользователем явно (*attribute propagation*) и проверка на совместимость всех атрибутов сигналов с блоками, которые их принимают; замещение всех виртуальных подсистем блоками, которые составляют их содержимое (*model hierarchy flattening*); определение порядка интегрирования уравнений блоков в модели (при этом создается упорядоченный список блоков – *blocks sorted list*); определение величины временного шага (*time step*) для блоков, у которых эта величина не задана явно (*time step propagation*). В результате компиляции по графической диаграмме модели создается исполняемый файл модели (*executable form*).

Следующий этап – этап, называемый *linking*. На этом этапе Simulink захватывает (аллоцирует) и инициализирует память, необходимую для хранения значений сигналов и состояний блоков. На этом же этапе на основе *blocks sorted list* определяется наиболее эффективный порядок вызовов методов блоков модели (создается *method execution list*). Пользователь при создании модели может влиять на порядок методов в *method execution list*, задавая приоритет того или иного блока. Методы блоков с более высоким приоритетом вызываются раньше.

Заключительный этап симуляции модели называется *simulation loop*. Он, в свою очередь, делится на две фазы: *loop initialization phase* и *loop iteration phase*. В первой фазе вычисляются начальные значения выходных сигналов и состояния модели. Эти вычисления выполняются один раз в момент запуска симуляции. Вторая фаза повторяется итеративно для каждого временного интервала, начиная с момента времени, заданного пользователем как начальный, и до момента окончания симуляции. На каждом шаге ите-

ративной фазы рассчитываются новые значения входных сигналов, состояния и выходных сигналов моделируемой системы. Среди стандартных блоков Simulink есть блоки, позволяющие показывать или записывать текущие значения сигналов, рассчитываемые на каждом шаге итеративной фазы.

Создание модели дискретной системы. Цифровой фильтр

Simulink является средой моделирования, интегрированной со средой научных расчетов MatLab. Первые шаги по созданию новой модели выполняются в среде MatLab.

1. Запустите MatLab.
2. Выберите рабочую директорию для модели. Пиктограмма, на которую следует нажать, чтобы появилось окно диалога для выбора рабочей директории указана на рис. 1 стрелкой.

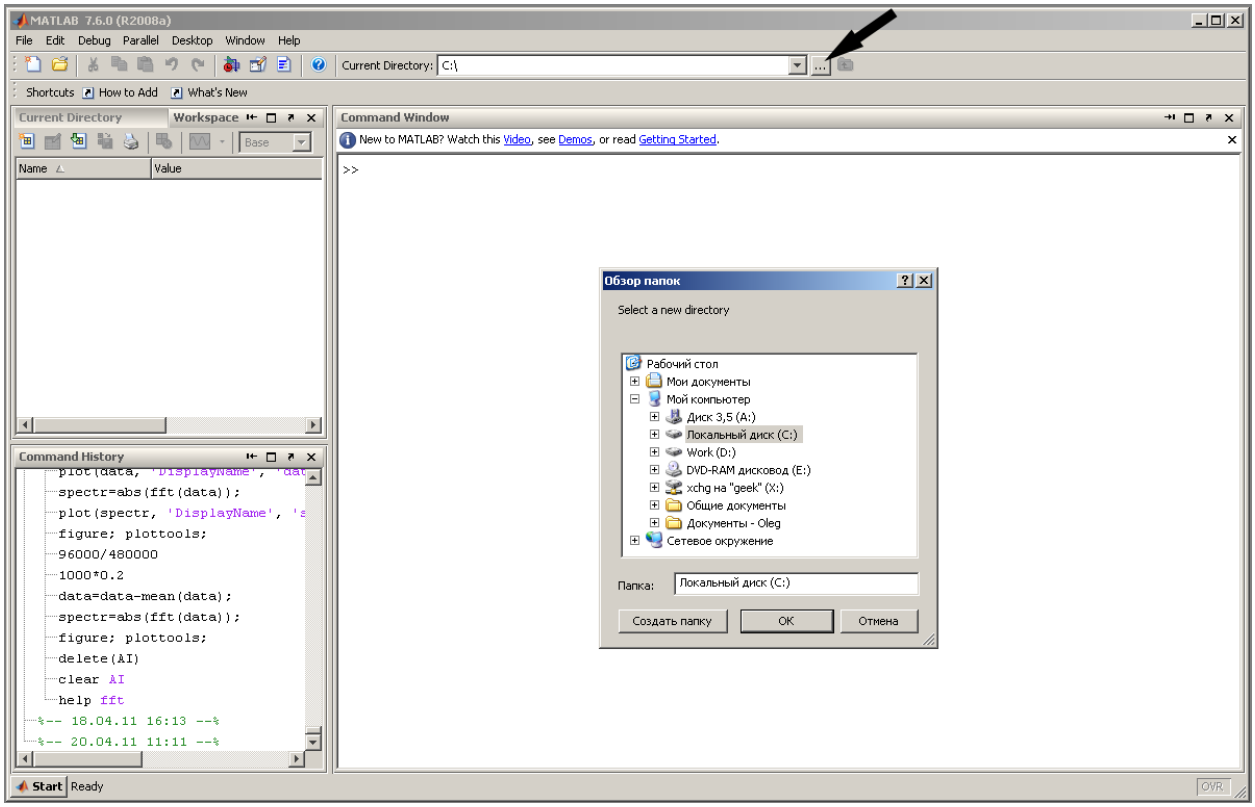


Рис. 1. Окно выбора рабочей директории программы MatLab

3. Запустите Simulink, нажав на пиктограмму на панели инструментов MatLab (рис. 2).

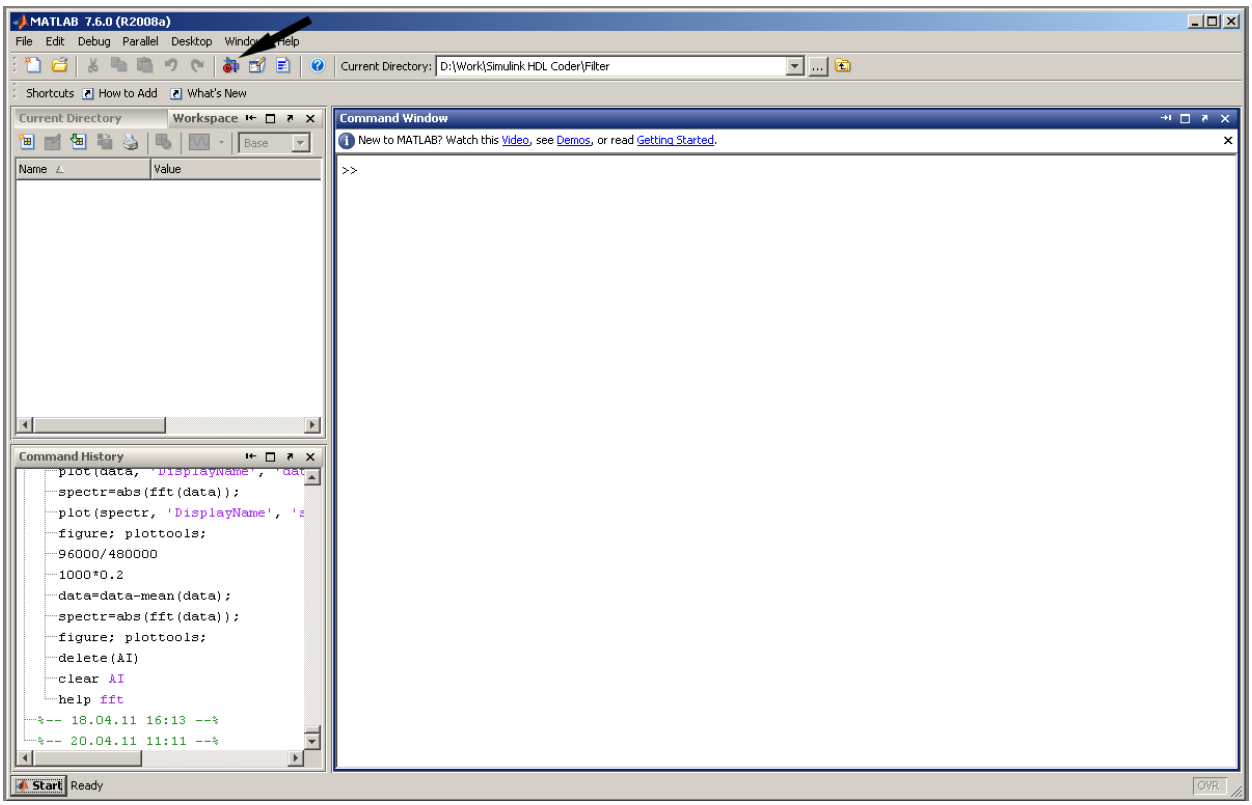


Рис. 2. Пиктограмма Simulink на панели инструментов MatLab

В результате откроется окно Simulink Library Browser (рис. 3), в панели инструментов которого находится пиктограмма создания новой модели.

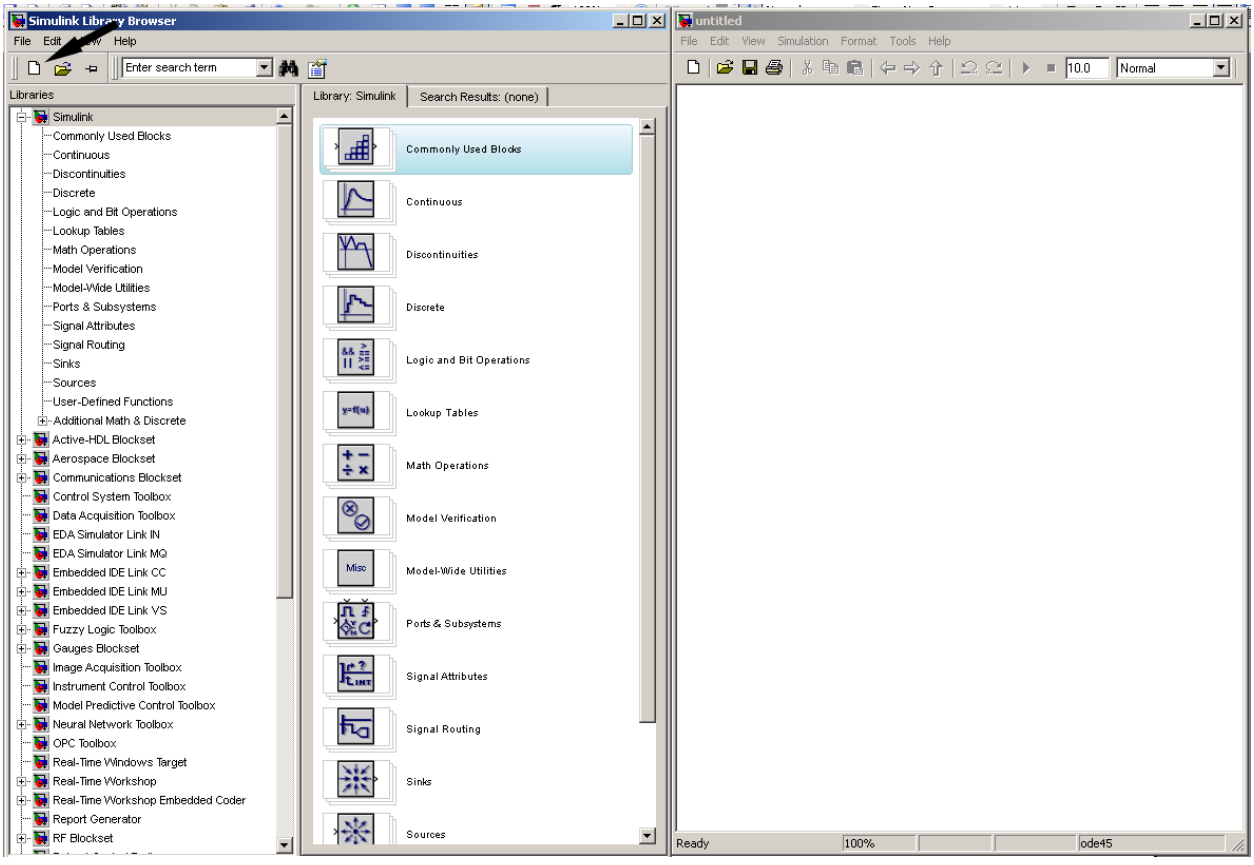


Рис. 3. Пиктограмма создания новой модели в панели инструментов Simulink Library Browser

После нажатия на эту пиктограмму откроется окно графического редактора модельных диаграмм. Прежде всего, необходимо сохранить новую модель, нажав на соответствующую пиктограмму в панели инструментов редактора.

4. Произведите настройку параметров модели. Для этого переключитесь в Command Window программы MatLab и наберите команду:

hdlsetup

По этой команде будет произведена настройка параметров модели, допускающих генерацию из нее в дальнейшем HDL-описания. После настройки параметров и сохранения модели можно приступать к созданию модельной диаграммы (при сохранении модели укажите имя MyFilter). Этот процесс заключается в перетаскивании блоков из Simulink Library Browser и соединении выходных и входных портов блоков сигналами. Далеко не все блоки, предоставляемые Simulink Library Browser, могут быть использованы в модели, предназначенной для генерации HDL. Список всех блоков, пригодных для этой цели, можно получить, набрав в Command Window программы MatLab команду:

hdllib

5. В разделе Discret в Simulink Library Browser выберите и перетащите в окно модели блок Unit Delay (рис.4). Размножьте блок в окне модели 8 раз (для этого при выделении блока в окне модели необходимо держать нажатой клавишу Ctrl)

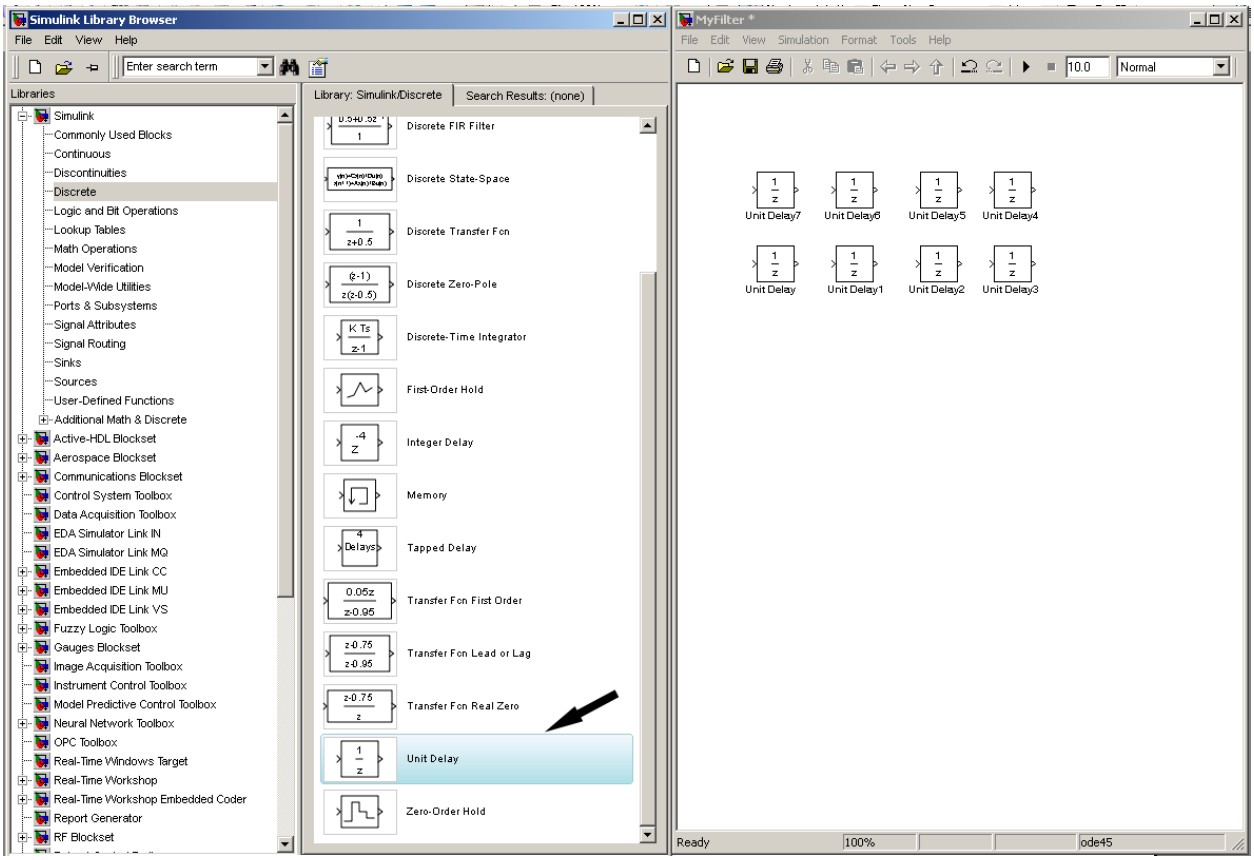


Рис. 4. Блок Unit Delay в разделе Discret программы Simulink Library Browser

6. Разверните 4 блока верхнего ряда на 180 градусов. Для этого выделите эти блоки и в выпадающем меню, доступном по нажатии правой кнопки «мыши», выберите позицию Format/Rotate block (рис. 5).

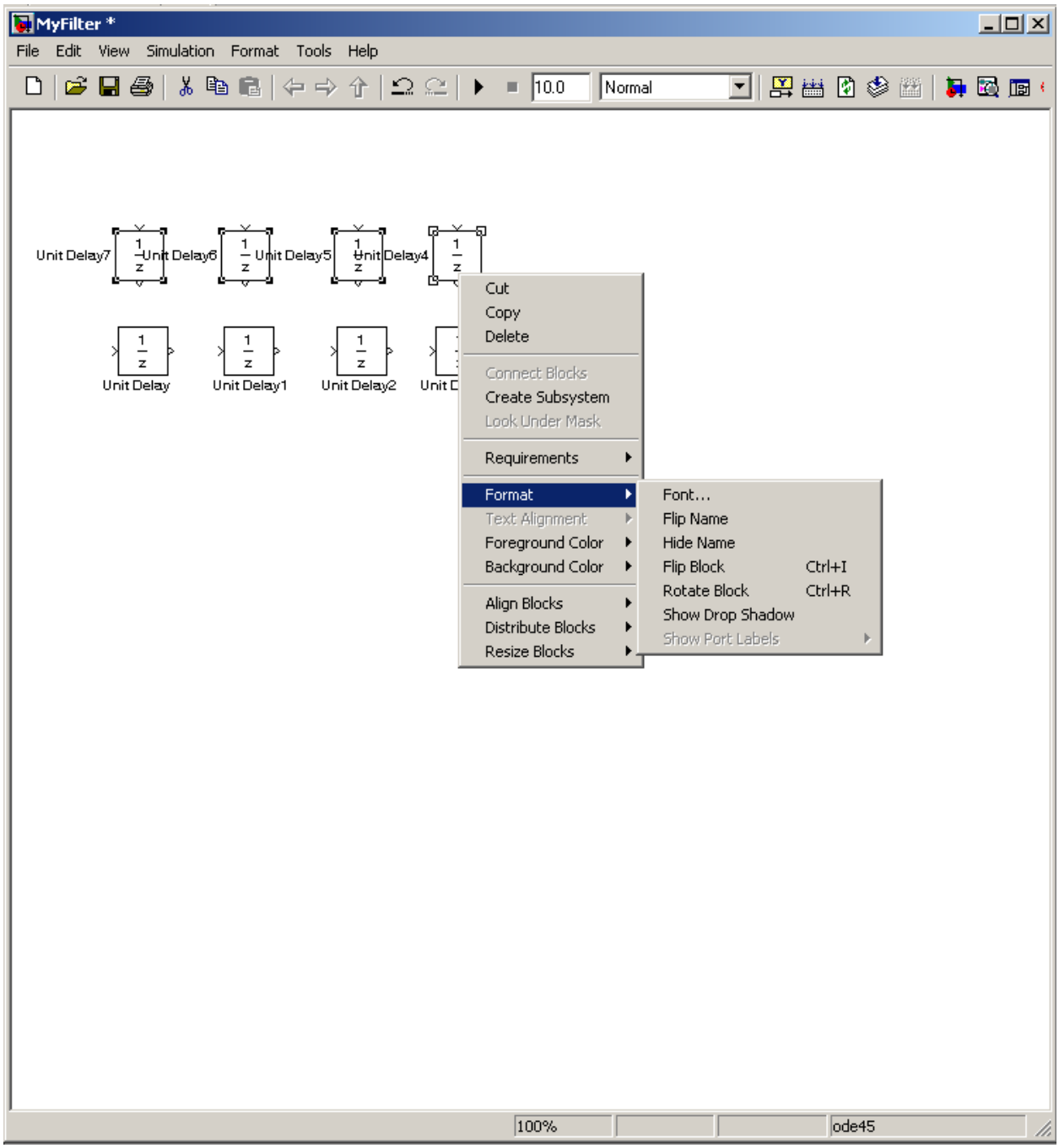


Рис. 5. При нажатии на правую кнопку «мыши» становится доступным выпадающее меню

7. Добавьте к модели блок Add из раздела Math Operations в Simulink Library Browser и размножьте его 7 раз. Из этого же раздела добавьте к модели блок Product и размножьте его 4 раза (рис. 6).

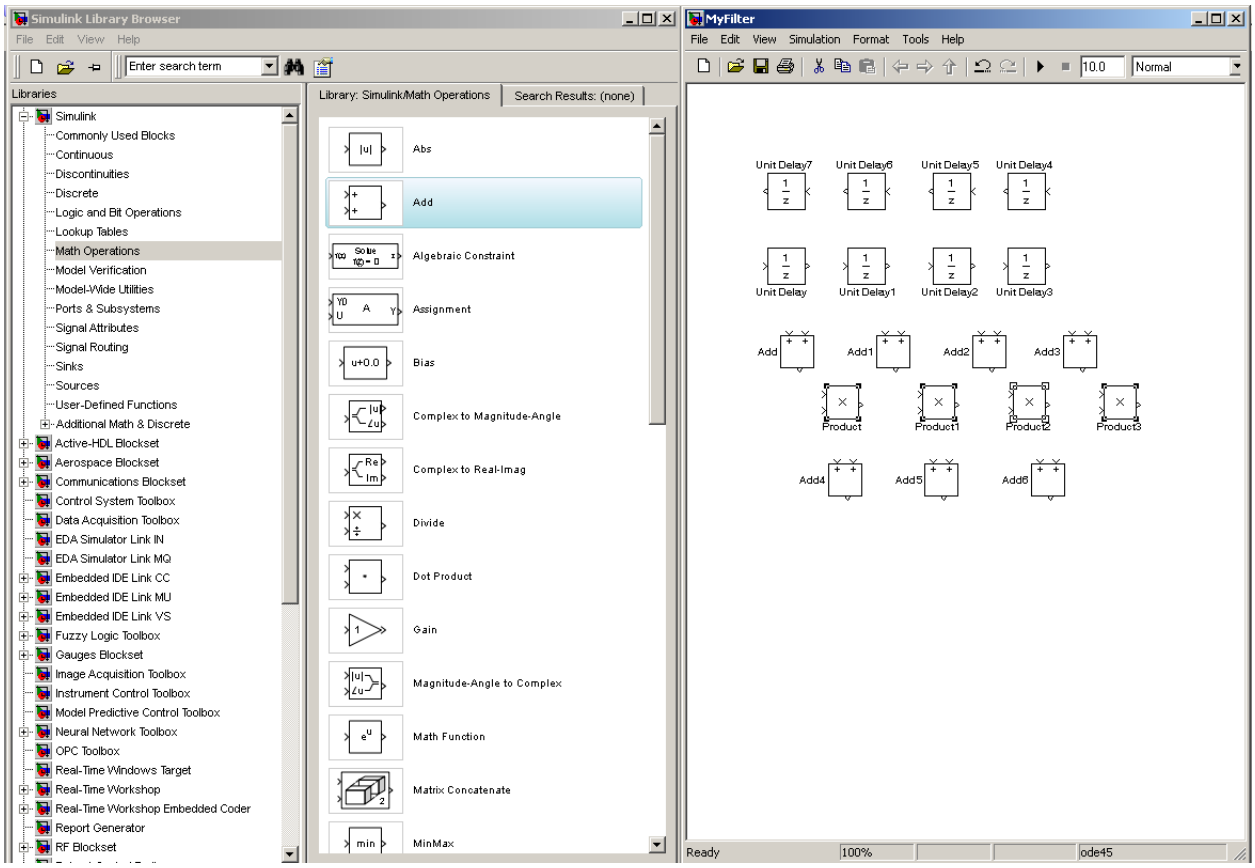


Рис. 6. Блок Add находится в разделе Math Operations в Simulink Library Browser

8. Начиная с выходного порта блока Unit Delay соедините порты модели сигналами, как показано на рис. 7. (Для соединения выходного порта блока с входным портом другого блока необходимо нажать на левую кнопку «мыши» на выходном порте и протянуть соединение, не отпуская левой кнопки «мыши», до входного порта. Для создания разветвлений сигнала, необходимо нажать на левую кнопку мыши в точке на сигнале, где создается разветвление, держа при этом нажатой клавишу Ctrl клавиатуры).

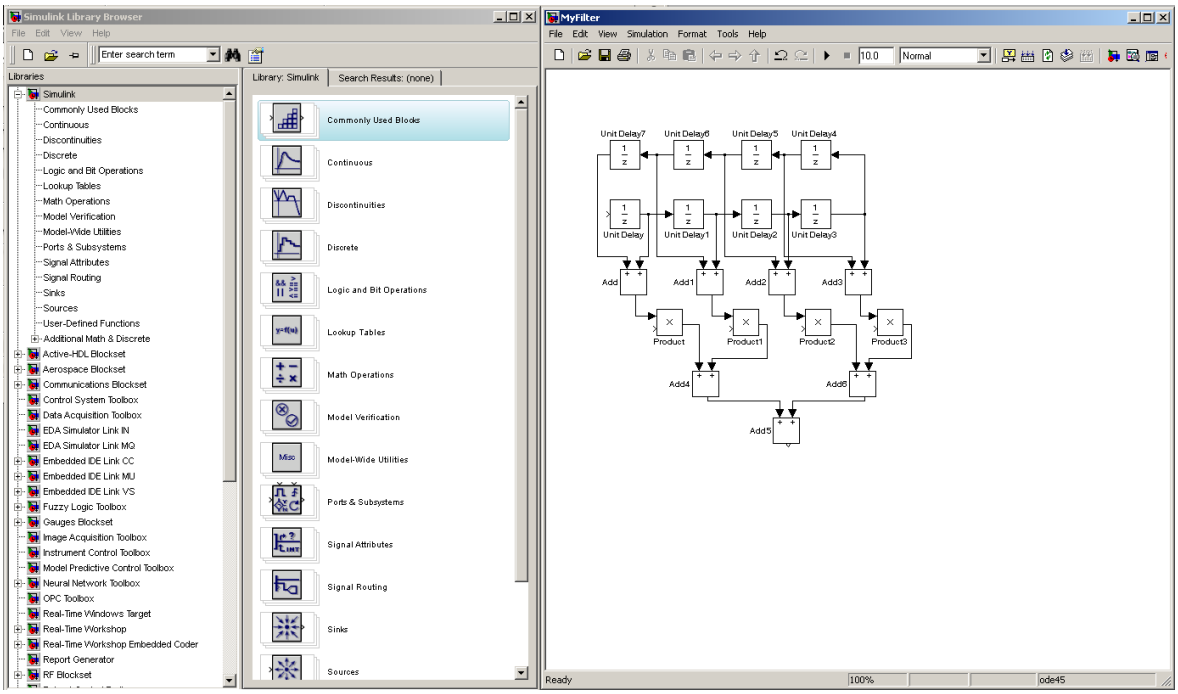


Рис. 7. Соединение блоков модели сигналами

9. Из раздела Sources в Simulink Library Browser добавьте к модели блок From Workspace. (Этот блок лучше добавить слева от существующих блоков модели. Если слева оказывается мало места, то существующие блоки можно легко сдвинуть. Для этого необходимо выделить все на диаграмме модели, обведя область с блоками и сигналами «мышью», или нажав комбинацию клавиш Ctrl+A. После этого, используя «мышь» или клавишу «стрелка вправо» клавиатуры можно сдвинуть блоки и сигналы модели на необходимое расстояние вправо).
10. Дважды щелкните по левой кнопке «мыши» над добавленным блоком From Workspace. Откроется окно диалога, позволяющее задать параметры блока (рис. 8). В строке Data введите следующее выражение (в этом выражении используется язык программирования MatLab):

[[0:2000].' cos(2.*pi.*[0:0.001:2].*(1+[0:0.001:2].*75)).'].

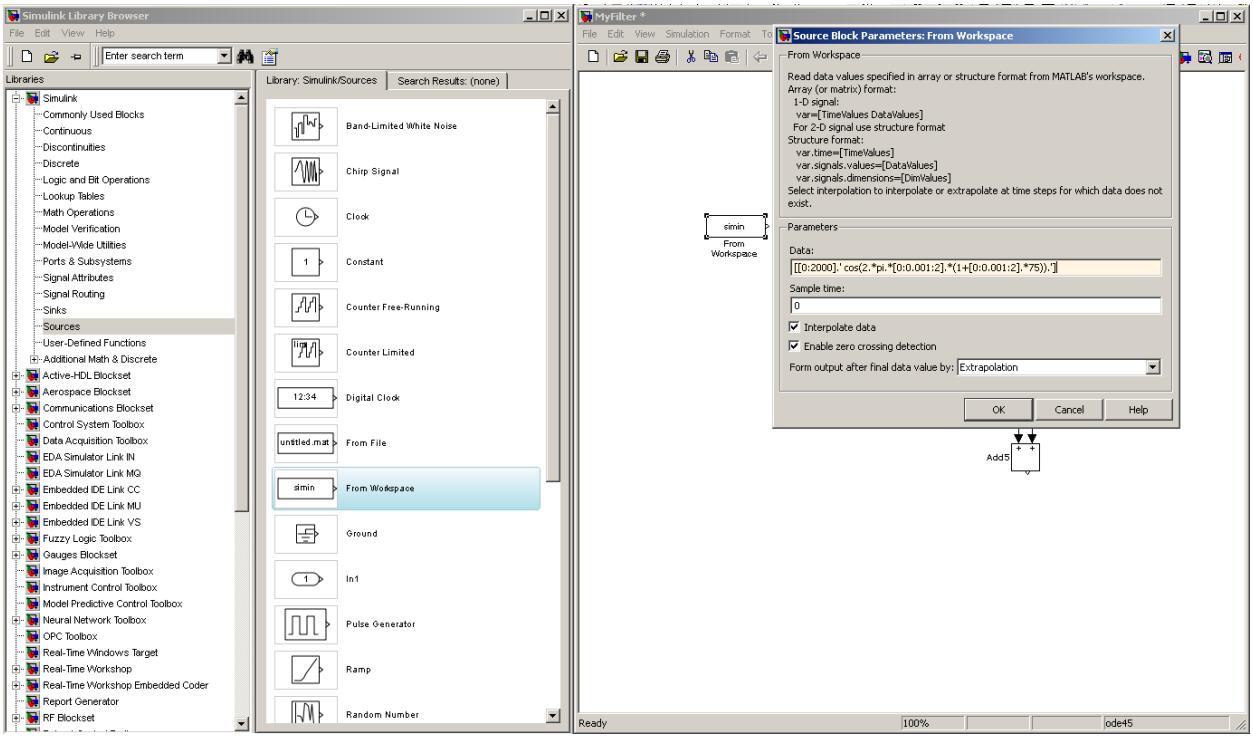


Рис. 8. Окно диалога для ввода параметров блока From Workspace откроется при двойном щелчке «мыши» по этому блоку после добавления его к модели

11. Добавьте блок Constant к модели из раздела Sources в Simulink Library Browser. Этот блок будет служить для задания одного из коэффициентов фильтра. В качестве параметра этого блока (дважды щелкнув по нему «мышью») задайте число -0.1339. На вкладке Signal Attributes в поле Output data type задайте fixdt(1,16,10).
12. Размножьте блок Constant 4 раза. В качестве параметра каждого нового блока задайте числа: -0.0838, 0.2026, 0.4064.
13. Добавьте блок Data Type Conversion из раздела Signal Attributes в Simulink Library Browser. В поле Ounput Data Type окна диалога параметров этого блока задайте fixdt(1,16,10).
14. Измените разрядность выходных сигналов умножителей (Product, Product1, Product2, Product3). Для этого двойным щелчком «мыши» на каждом из блоков откройте окно диалога для задания параметров блока. Перейдите на вкладку Signal Attributes и в поле Output data type укажите fixdt(1,16,10). (В предлагаемой схеме на вход каждого из умножителей поступает по два сигнала, один из которых является 16-ти разрядным, а второй 17-ти разрядным.

Результат умножения, таким образом, окажется 33-х разрядным. Simulink не допускает генерации HDL кода по модели, в которой есть более чем 32-х разрядные сигналы)

15. Добавьте блок Scope из раздела Sinks в Simulink Library Browser. Двойным щелчком «мыши» по этому блоку откройте окно Scope и нажмите на пиктограмму Parameters. В появившемся окне в поле Number of Axes установите 2.
16. Соедините добавленные блоки сигналами, как показано на рис. 9.
17. Установите время симуляции в панели инструментов модели равным 2000.

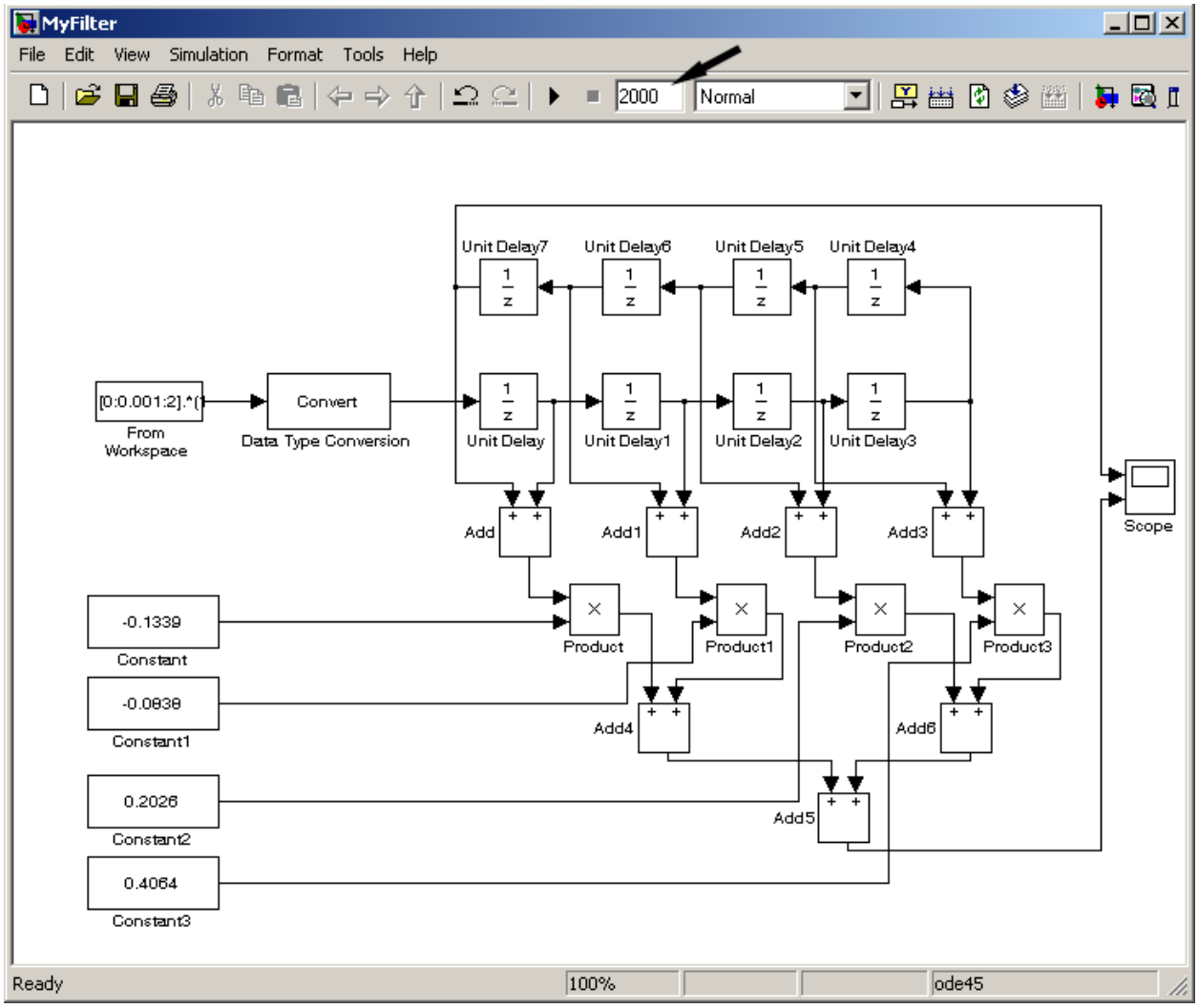


Рис. 9. Стрелкой на рисунке указано место, где можно задать время симуляции модели

18. Запустите моделирование, нажав на кнопку Start Simulation на панели инструментов окна модели.

После завершения симуляции в окне Scope будут отображены два графика: верхний – входной сигнал фильтра, задержанный на 8 тактов (временных шагов) работы модели; нижний – результат фильтрации входного сигнала. При необходимости размеры окна Scope легко меняются при помощи «мыши». Щелчок правой кнопкой «мыши» в окне Scope открывает доступ к выпадающему меню. Позиция Autoscale этого меню позволяет автоматически подстраивать масштабы сигнала на каждом графике.

Simulink HDL Coder

Расширение среды моделирования Simulink, называемое Simulink HDL Coder, позволяет выполнять автоматическую генерацию кода на одном из языков описания

аппаратуры (Verilog или VHDL) для цифрового устройства, модель которого создана в Simulink. Это средство в ряде случаев может существенно повысить скорость разработки и отладки цифровых устройств, позволяя разработчику сосредотачиваться на алгоритмах обработки сигналов в разрабатываемом цифровом устройстве и его архитектуре, а не на кодировании этих решений на языке описания аппаратуры.

Процесс генерации VHDL кода по готовой модели очень прост. Во-первых, разработчик должен запустить утилиту Simulink HDL Coder compatibility checker, которая проверит совместимость модели устройства с HDL Coder. Второй шаг – собственно генерация VHDL-описания модели. Третий шаг – генерация VHDL-программы TestBench для тестирования сгенерированного кода.

Здесь необходимо отметить несколько моментов. Прежде всего, если посмотреть на диаграмму модели цифрового фильтра (рис. 9), то станет ясно, что далеко не все блоки модели относятся непосредственно к реализации разработанного фильтра. Так, например, блоки From Workspace, Data Type Conversion, Scope, Constant, Constant1 и т.д. предназначены для моделирования входного (фильтруемого) сигнала, отображения результата фильтрации на графике, задания коэффициентов фильтра. Эти блоки, таким образом, моделируют окружение (среду), в котором функционирует разработанный фильтр. Если говорить в терминах языка описания цифровой аппаратуры, то эти блоки относятся к описанию TestBench. Очевидно, что перед генерацией VHDL-кода для разработанного фильтра, необходимо отделить блоки модели, относящиеся непосредственно к архитектуре фильтра, от тех, что моделируют условия его работы. Такое разделение в Simulink можно осуществить с помощью блока Subsystem и специального файла, называемого Control file.

Объединение набора блоков в Subsystem в Simulink выполнить очень просто. Выполним это в модели цифрового фильтра.

1. С помощью «мыши» выделите в модели те блоки, которые относятся непосредственно к реализации архитектуры фильтра (это все блоки модели за исключением, как уже указывалось, блоков From Workspace, Data Type Conversion, Scope, Constant, Constant1, Constant2, Constant3).
2. Не снимая выделения, нажмите на правую кнопку «мыши» и из выпадающего меню выберите позицию Create Subsystem (рис. 10).

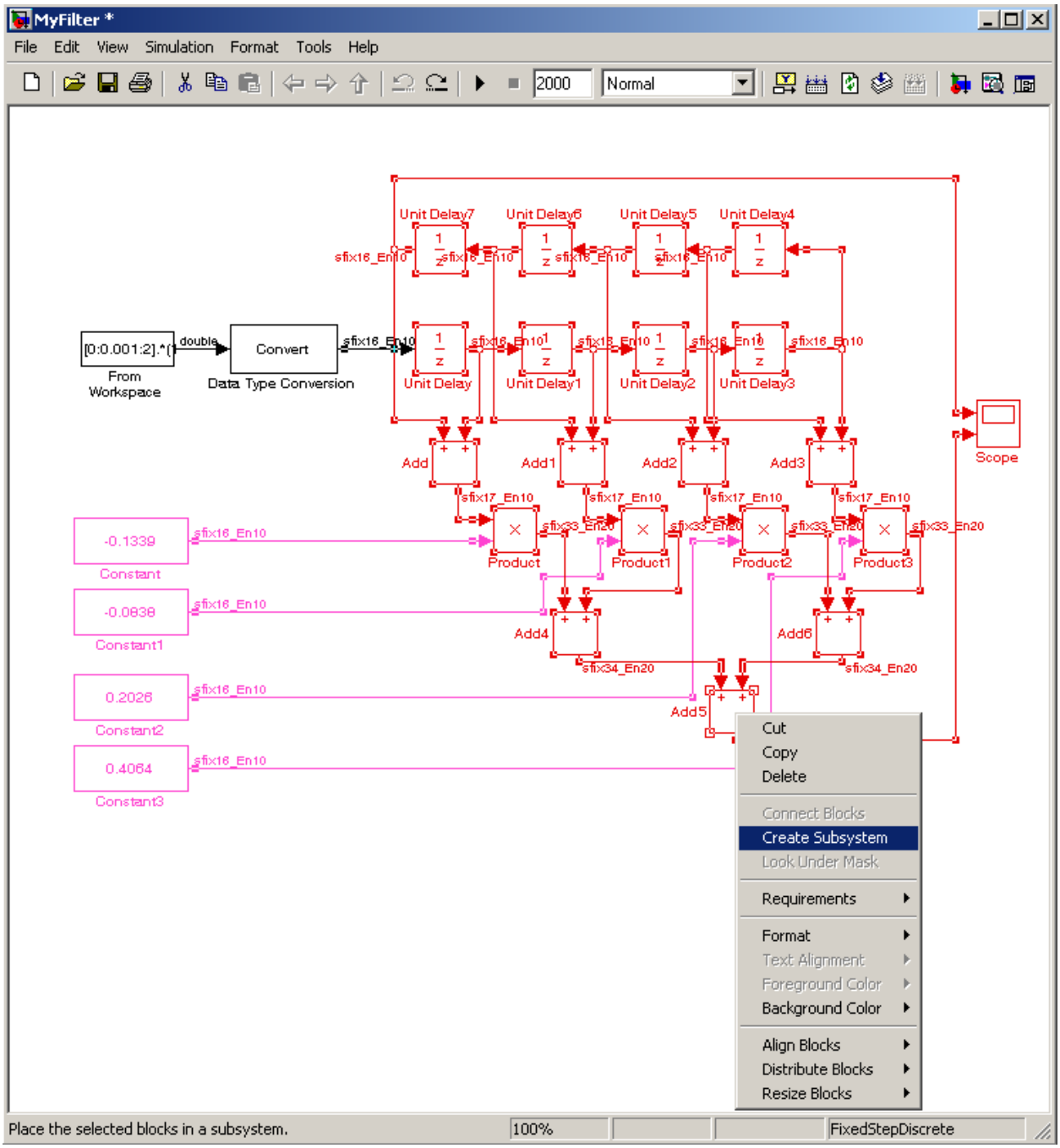


Рис. 10. Для объединения блоков в подсистему выберите из контекстного меню, появляющегося при нажатии на правую кнопку «мыши», позицию Create Subsystem

В результате, выделенные блоки будут объединены в подсистему, а в диаграмме модели они будут заменены одним блоком с именем Subsystem. Щелчок левой кнопкой «мыши» по надписи Subsystem позволяет изменить имя подсистемы (например, на Digital Filter). Двойной щелчок левой кнопки «мыши» по блоку подсистемы открывает содержимое подсистемы в отдельном окне. Отметим, что входные и выходные порты подсистемы имеют стандартные названия In1, In2, …, Out1, Out2, … . Их можно поменять на более информативные названия, щелкнув левой кнопкой «мыши» в поле имени порта. Осмысленные названия подсистем и портов в модели существенно повышают ее удобочитаемость. (При необходимости, опять же для повышения читаемости диаграммы модели, размеры блока подсистемы можно поменять с помощью «мыши». Можно также повернуть блок подсистемы, используя команду Format/Rotate контекстного меню, выпадающего при нажатии на правую кнопку «мыши»).

3. Поменяйте названия портов подсистемы Digital Filter так, чтобы модель выглядела как на рис. 11.

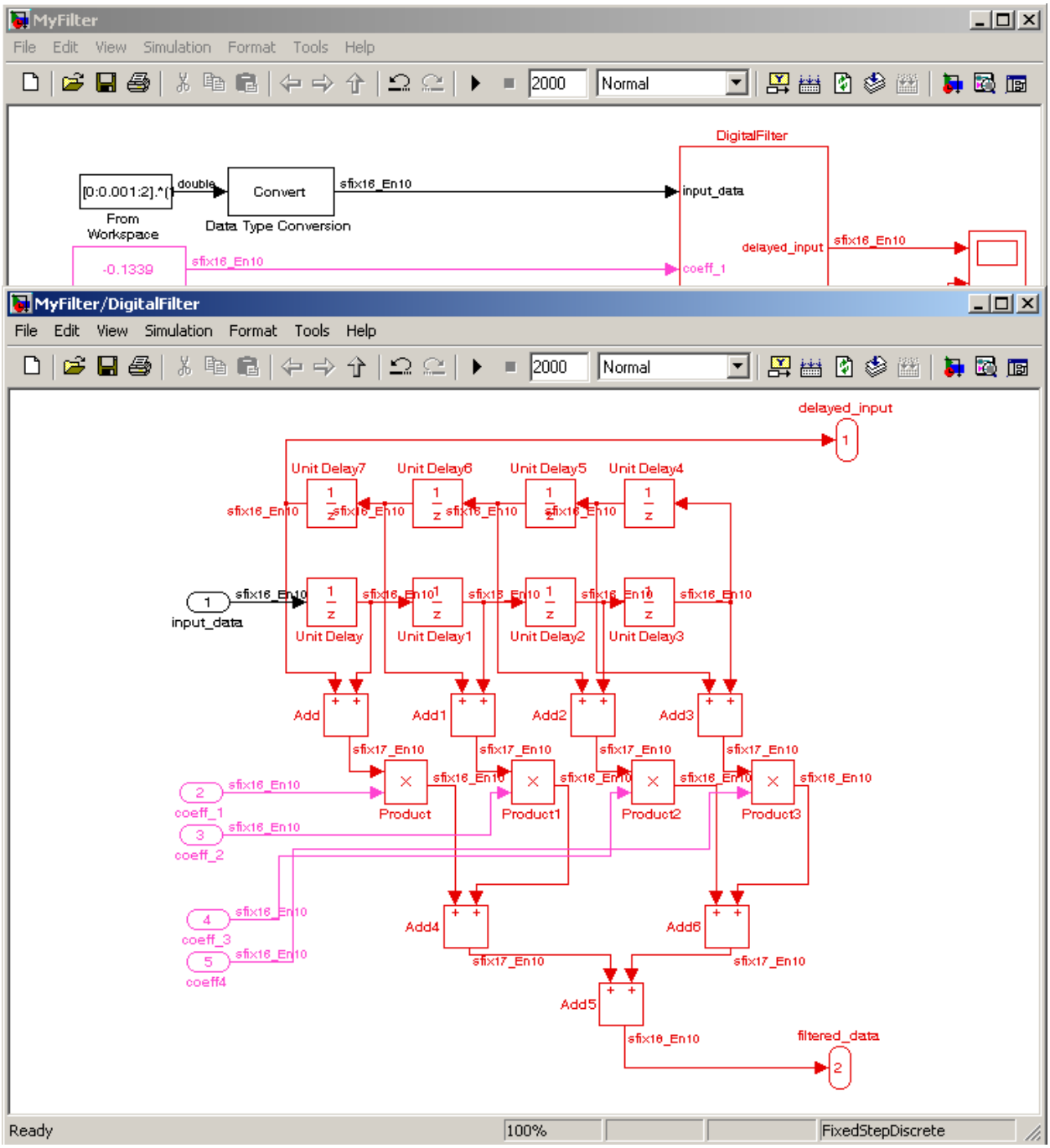


Рис. 11. Поменяйте имена входных и выходных портов подсистемы для повышения читаемости диаграммы модели

Следующий шаг, необходимый для правильной генерации HDL-кода – написание Control File. Наиболее просто создать правильный Control File используя скрипт, поставляемый вместе с расширением HDL Coder.

4. В меню Simulation окна модели выберите позицию Configuration Parameters. В левой части открывшегося окна диалога (дерево Select) выберите позицию HDL Coder. Из выпадающего списка Generate HDL for выберите MyFilter/DigitalFilter. Из выпадающего списка Language выберите VHDL (рис. 12).

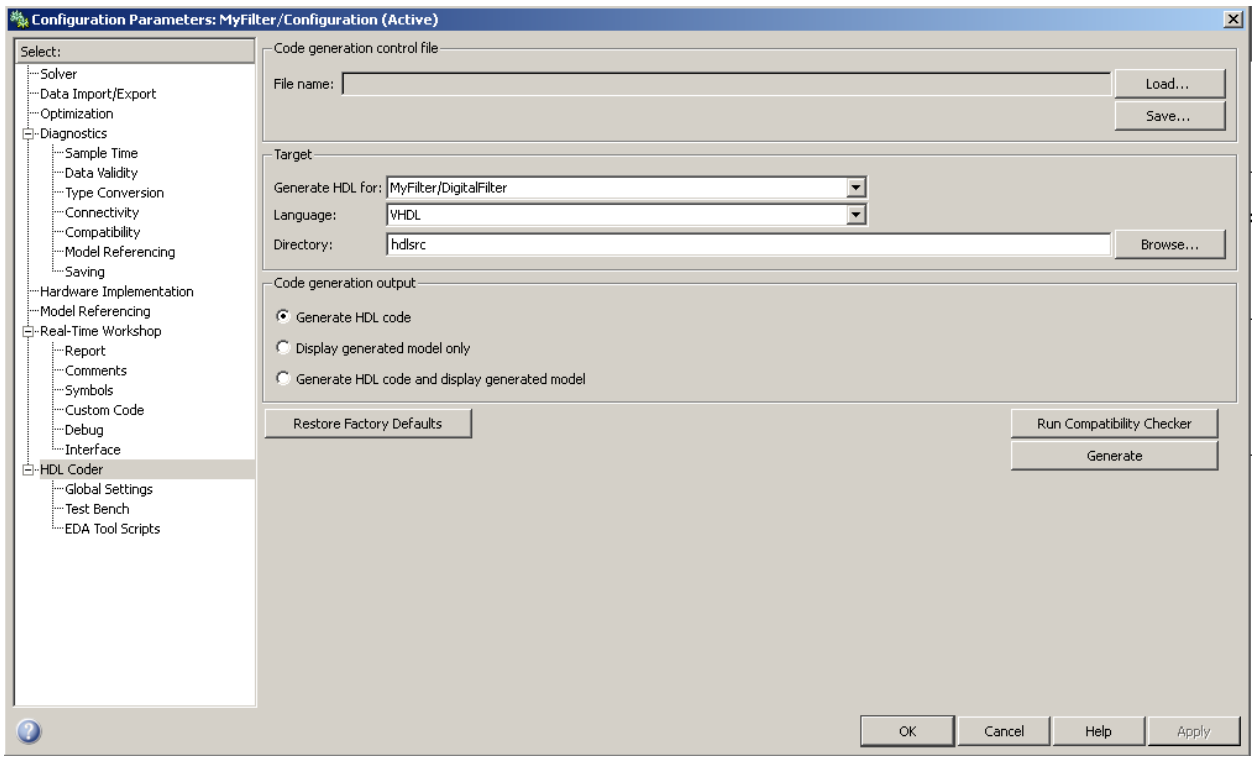


Рис. 12. Установите правильные параметры генерации HDL-кода

5. В Command Window среды MatLab наберите команду:

hdlnewcontrolfile(block),

где аргумент block – полный путь к подсистеме, представляющей собой модель цифрового устройства. В нашем случае в качестве аргумента необходимо набрать:

hdlnewcontrolfile('MyFilter/DigitalFilter')

(Путь к подсистеме указывается в одинарных кавычках, элементы пути разделяются символом «/»).

В результате выполнения скрипта в текстовом редакторе MatLab будет создан новый текстовый файл с именем controlfilename. При сохранении этого файла его имя лучше поменять, например, на MyFilterControlFile.

6. Укажите созданный MyFilterControlFile в качетве Code Generation Control File. Для этого переключитесь в окне диалога Configuration Parameters нажмите на кнопку Load и в открывшейся директории выберите для загрузки MyFilterControlFile (рис. 13).

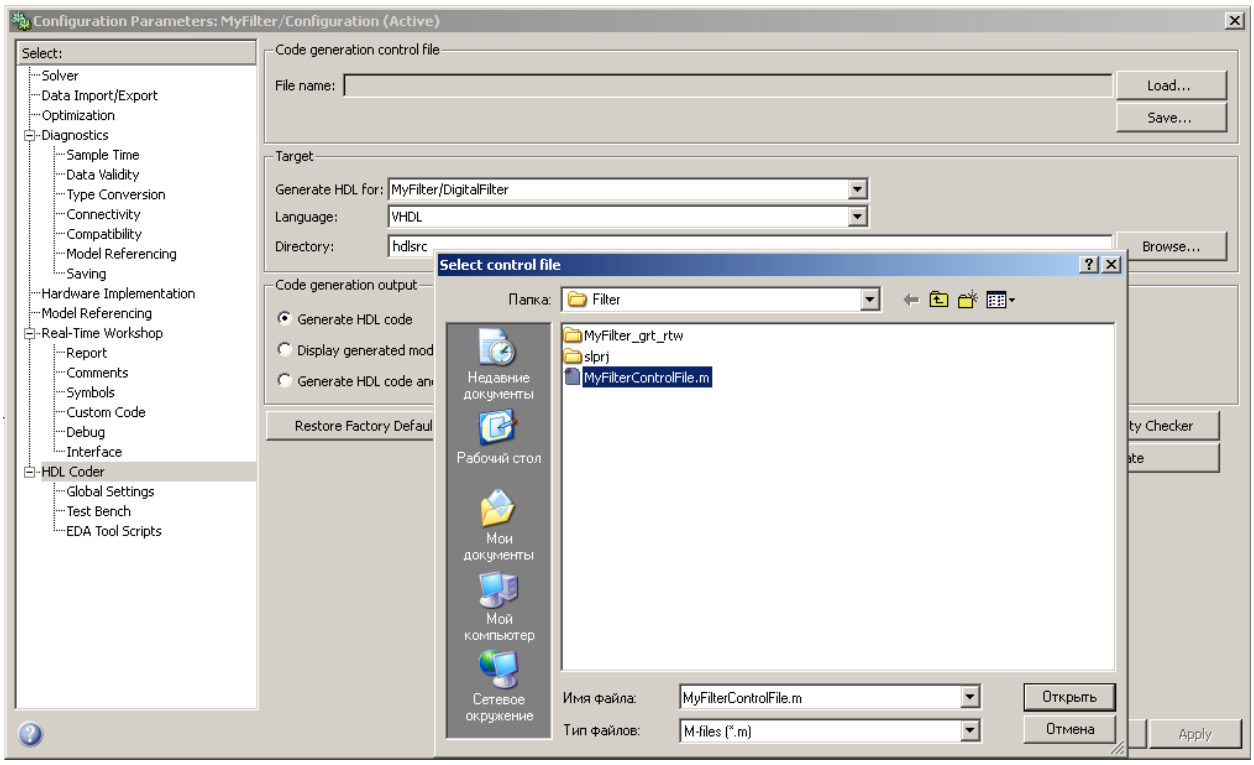


Рис. 13. Выберите сгенерированный MyFilterControlFile в качестве Code generation control file

Модель готова к генерации VHDL-кода.

7. В окне диалога Configuration Parameters нажмите на кнопку Run Compatibility Checker.

Утилита Compatibility Checker выполняет проверку модели на совместимость с Simulink HDL Coder. По результатам проверки выполняется генерация HTML-файла, который открывается в WEB-браузере системы (рис. 14).

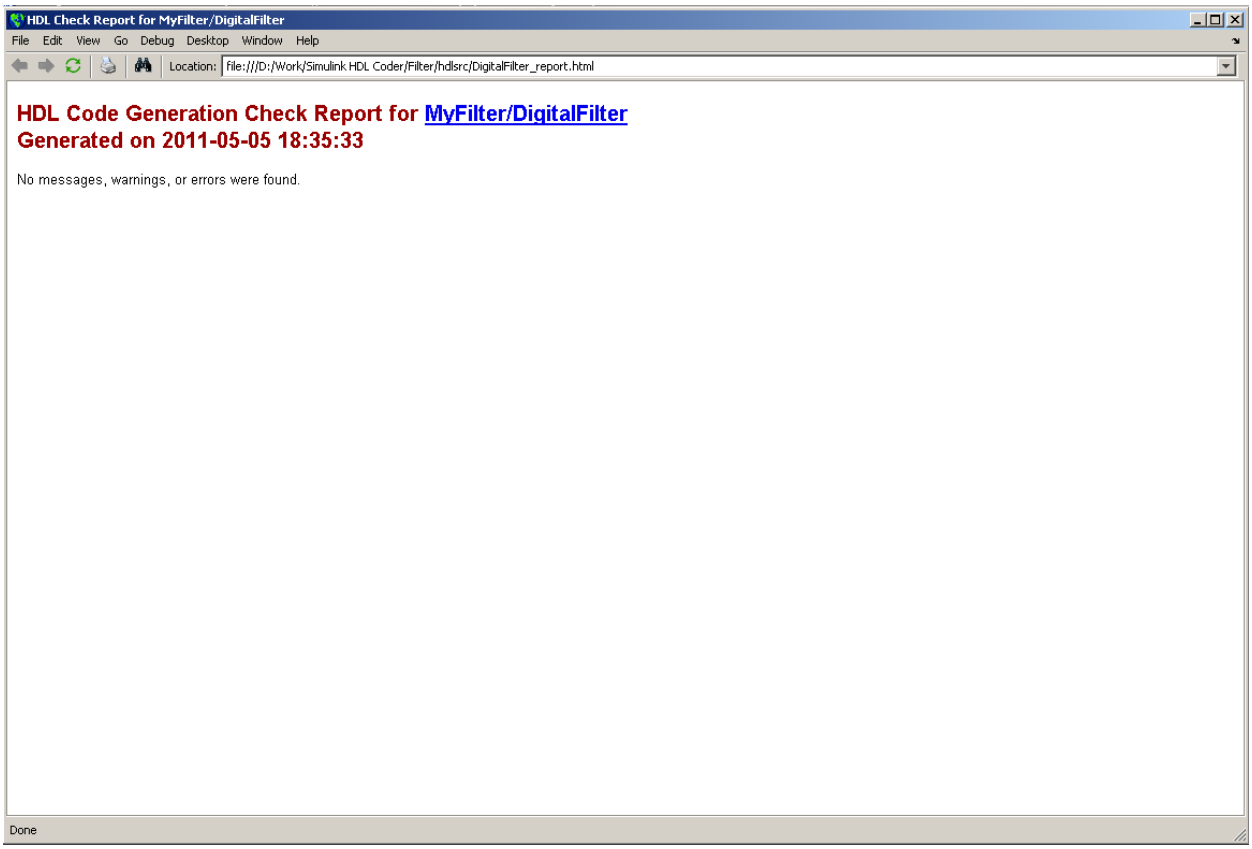


Рис. 14. Результаты проверки отображаются в WEB-браузере

8. Выполните генерацию VHDL кода по модели цифрового фильтра. Для этого нажмите на кнопку Generate в окне диалога Configuration Parameters.

В окне Command Window программной среды MatLab после генерации кода появится сообщение, указывающее на успешность выполнения процесса генерации и полный путь к сгенерированному файлу. Отметим, что количество сгенерированных файлов может варьироваться в зависимости от сложности модели.

Для генерации TestBench для модели цифрового фильтра необходимо выполнить следующие шаги.

9. В «дереве» выбора Select диалогового окна Configuration Parameters выберите TestBench.
10. Нажмите на кнопку Generate Test Bench

В результате запустится генерация HDL-кода для тестирующей программы TestBench. По окончании генерации в окне Command Window программной среды MatLab будет указан путь к сгенерированному HDL-файлу.

Несколько слов о control file. Этот файл позволяет производить настройку процесса генерации HDL-кода по модели. Причем, настройку можно производить как для модели в целом, так и для отдельных блоков и подсистем. Настройка выполняется с помощью команды forEach, помещаемой в control file для каждого блока, генерация кода для которого нуждается в настройке. Наиболее просто создать правильный вызов команды forEach с помощью скрипта hdlnewforeach, поставляемого в составе расширения Simulink HDL Coder. Подробности использования команды forEach в control file можно изучить по документации к Simulink HDL Coder.

Работа с битами

Одна из часто встречающихся задач, возникающих при разработке цифровых устройств, связана с разбором битовых полей в потоке данных, передаваемых побайтно. Примером такой задачи может служить задача демультиплексирования транспортного

потока MPEG-2. Данные в потоке объединены в пакеты длиной 188 байт. Начало каждого пакета отмечается синхробайтом со значением 0x47, за которым следует заголовок пакета. Заголовок состоит из битовых полей различной разрядности с общей длиной 4 байта. Значения битовых полей в заголовке определяют набор операций (алгоритм) необходимый для демультиплексирования данного пакета или других пакетов.

Не углубляясь в структуру потока данных стандарта MPEG-2, разберем на примерах возможные варианты решения следующих двух задач:

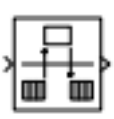
1. выделить из четырех последовательно поступающих байтов (4-х байтное число имеет разрядность 32) битовые поля с номерами разрядов 0 – 3; 4 – 10; 11 – 12; 13 – 20; 21 – 31;
2. описать вентиль, управляемый значениями одноразрядного сигнала; если управляющий сигнал имеет значение 1, на выход передаются данные, поступающие на вход вентиля; если управляющий сигнал имеет значение 0, на выходе вентиля сохраняется последнее переданное значение.

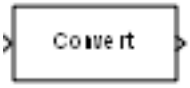
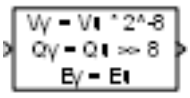
Рассмотрим первую задачу.

1. Создайте на диске директорию ExtractBits, запустите MatLab и укажите ExtractBits в качестве рабочей директории (Current Directory).
2. Запустите Simulink Library Browser и создайте новую модель. Сохраните ее под именем ExtBits.
3. Настройте параметры модели, запустив в Command Window среды MatLab скрипт

hdlsetup

4. В таблице ниже приведены блоки с кратким описанием их функциональности, которые необходимо поместить в диаграмму модели из Simulink Library Browser.

Символ и название	Категория в Simulink Library Browser	Функциональность и настройки
 Counter Free-Running	Sources	Источник байтового потока. В окне диалога Source Block Parameters в поле Number of bits установите 8. В поле Sample time установите 1.
 Taped Delay	Discrete	Накапливает значения входного сигнала заданное количество тактов. Выходной сигнал представляет собой вектор из накопленных значений. В поле Sample Time окна диалога Function Block Parameters установите -1. В поле Number of Delays – значение 4. В выпадающем списке Order output vector starting with выберите позицию Oldest
 Rate Transition	Signal Attributes	Согласовывает скорости работы входного и выходного портов. В поле Initial Condition окна диалога Function Block Parameters установите 0. В списке Output port sample time options выберите позицию Multiple of input port sample time. В поле Sample time multiple установите 4.

Символ и название	Категория в Simulink Library Browser	Функциональность и настройки
 Data Type Conversion	Signal Attributes	Осуществляет преобразование типа сигнала. В поле Output Data Type окна диалога Function Block Parameters установите uint32.
 Demux	Signal Routing	Разделяет векторный сигнал на отдельные скалярные значения. В поле Number of Outputs окна диалога Function Block Parameters установите 4.
 Shift Arithmetic	Logic and Bit Operations	Осуществляет битовый сдвиг (умножение на степень 2) входного сигнала. В поле Number of bits to shift right окна диалога Function Block Parameters установите -24.
 Bitwise Operator	Logic and Bit Operations	Выполняет побитовую операцию (какую-либо) над входными сигналами. Снимите флажок Use bit mask окна диалога Function Block Parameters. В выпадающем списке Operator выберите OR. В поле Number of input ports установите 4.
 Extract Bits	Logic and Bit Operations	Выделяет из входного сигнала заданные разряды. В выпадающем списке Bits to extract окна диалога Function Block Parameters выберите Range of bits. В поле Bit indices установите [0 3].
 Display	Sinks	Выводит на экран значения входного сигнала. В выпадающем списке Format окна диалога Function Block Parameters выберите binary.

5. Размножьте необходимые блоки и соедините их сигналами так, чтобы диаграмма модели выглядела как на рис. 14.

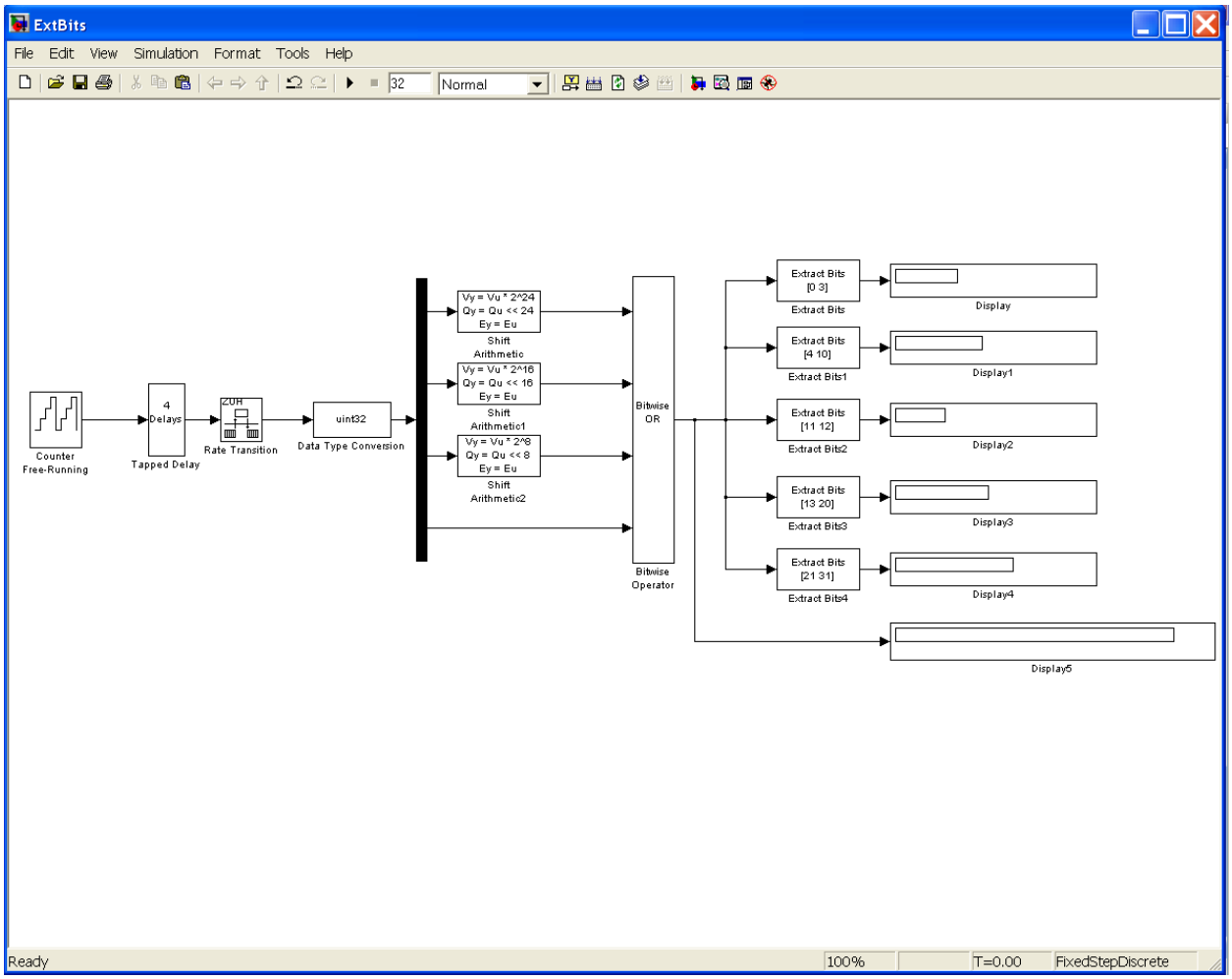


Рис. 7. Диаграмма модели

6. Поменяйте параметры блоков Shift Arithmetic1 (установите величину сдвига равной -16) и Shift Arithmetic2 (установите величину сдвига равной -8).
7. Поменяйте индексы извлекаемых битов в параметрах блоков Extract Bits1 (установите индексы [4 10]), Extract Bits2 (установите индексы [11 12]), Extract Bits3 (установите индексы [13 20]), Extract Bits4 (установите индексы [21 31]).
8. Скомпилируйте модель. Для этого нажмите на кнопку Incremental build в панели инструментов окна модели

Теперь выполним проверку работы модели. Это наиболее удобно сделать, запустив модель в пошаговом режиме, используя Simulink Debugger.

9. Запустите Simulink Debugger, нажав на кнопку Debug в панели инструментов окна модели.
10. На панели Break Points в поле Break at time установите значение 1 (рис. 15).

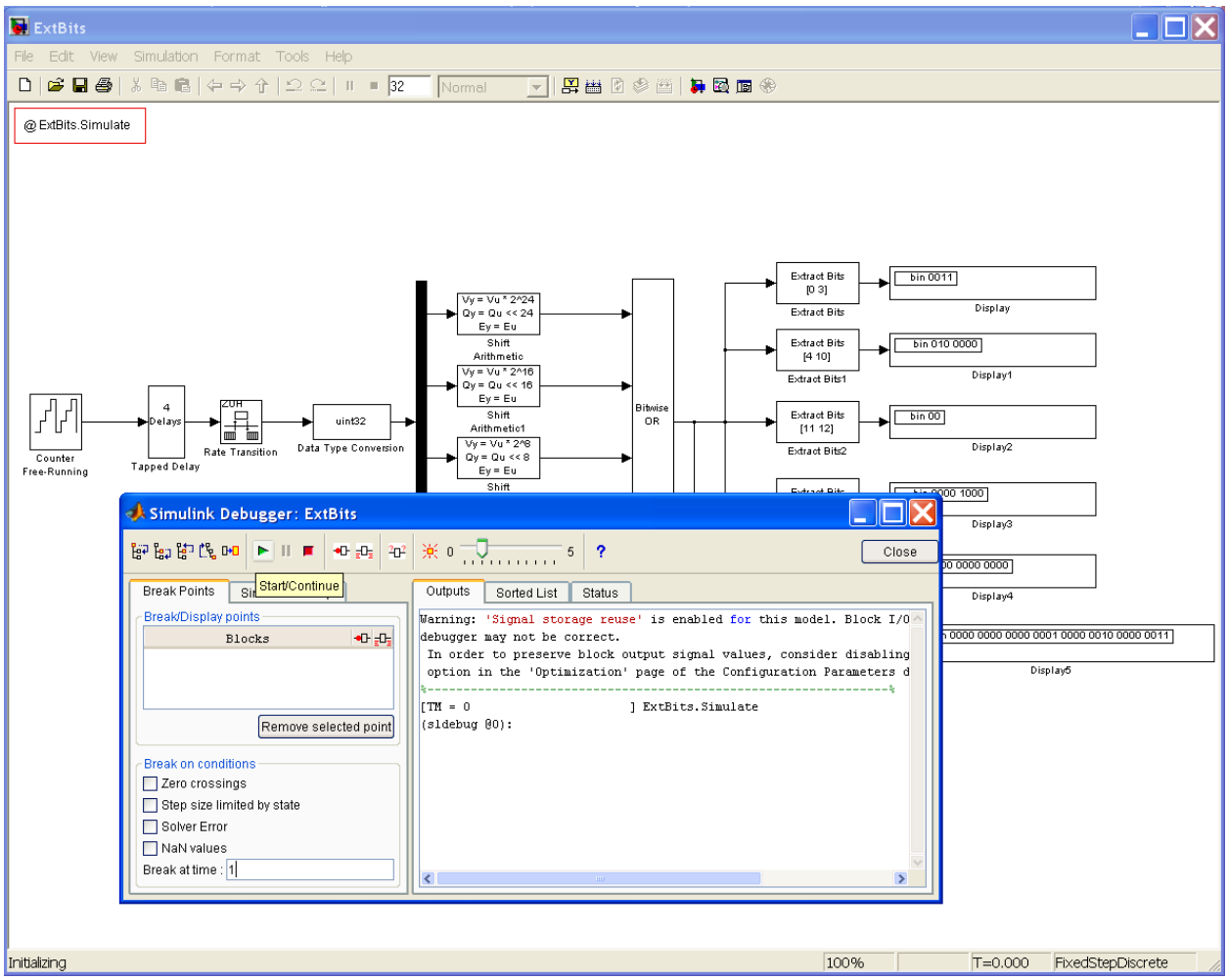


Рис. 15. Проверку работы модели наиболее удобно проводить в пошаговом режиме с помощью Simulink Debugger

11. Нажмите на кнопку Start/Continue панели инструментов окна Simulink Debugger.
12. Нажимая на кнопку Go to first method at start of next time step, выполняйте Simulation loop модели в пошаговом режиме.

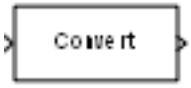
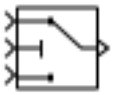
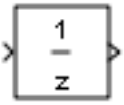
Можно повысить информативность пошагового выполнения модели, если нажать на кнопку Show port values when hovering. Если эта кнопка нажата, то при наведении курсора «мыши» на какой-либо блок модели на экране отображаются значения сигналов на выходных портах данного блока. Для генерации HDL-кода по созданной модели необходимо объединить все блоки модели за исключением источника данных (блок Counter Free-Running) и блоков Display, обеспечивающих вывод значений на экран, в подсистему. После указания имени созданной подсистемы в поле Generate HDL for в окне диалога Configuration Parameters модели, можно выполнить генерацию VHDL-кода для созданного устройства и для тестирующей программы Testbench.

Перейдем к выполнению второй задачи.

1. Создайте на диске директорию Gate, запустите MatLab и укажите Gate в качестве рабочей директории (Current Directory).
2. Запустите Simulink Library Browser и создайте новую модель. Сохраните ее под именем Gate.
3. Настройте параметры модели, запустив в Command Window среды MatLab скрипт

hdlsetup

В таблице ниже приведены блоки с кратким описанием их функциональности, которые необходимо поместить в диаграмму модели из Simulink Library Browser.

Символ и название	Категория в Simulink Library Browser	Функциональность
 Counter Free-Running	Sources	Источник байтового потока. В окне диалога Source Block Parameters в поле Number of bits установите 8. В поле Sample time установите 1.
 Pulse Generator	Sources	Источник управляющего сигнала. В окне диалога Source Block Parameters в списке Pulse type выберите Sample based, в списке Time – Use simulation time. В поле Amplitude установите значение 1, в поле Period – значение 10, в поле Pulse width – 5.
 Data Type Conversion	Signal Attributes	Осуществляет преобразование типа сигнала. В поле Output Data Type окна диалога Function Block Parameters установите boolean.
 Bitwise Operator	Logic and Bit Operations	Выполняет побитовую операцию (какую-либо) над входными сигналами. В выпадающем списке Operator окна диалога Function Block Parameters выберите NOT.
 Switch	Signal Routing	Управляемый переключатель. Под управлением второго порта коммутирует на выход сигналы первого или третьего портов. В выпадающем списке Format окна диалога Criteria for passing first input выберите u2~=0
 Unit Delay	Discret	Запоминает значение входного сигнала на один такт. На выходе значения входного сигнала появляются с задержкой в один такт.
 Display	Sinks	Выводит на экран значения входного сигнала

4. Разместите блоки в окне диаграммы модели и соедините их сигналами так, чтобы диаграмма выглядела как на рис. 16

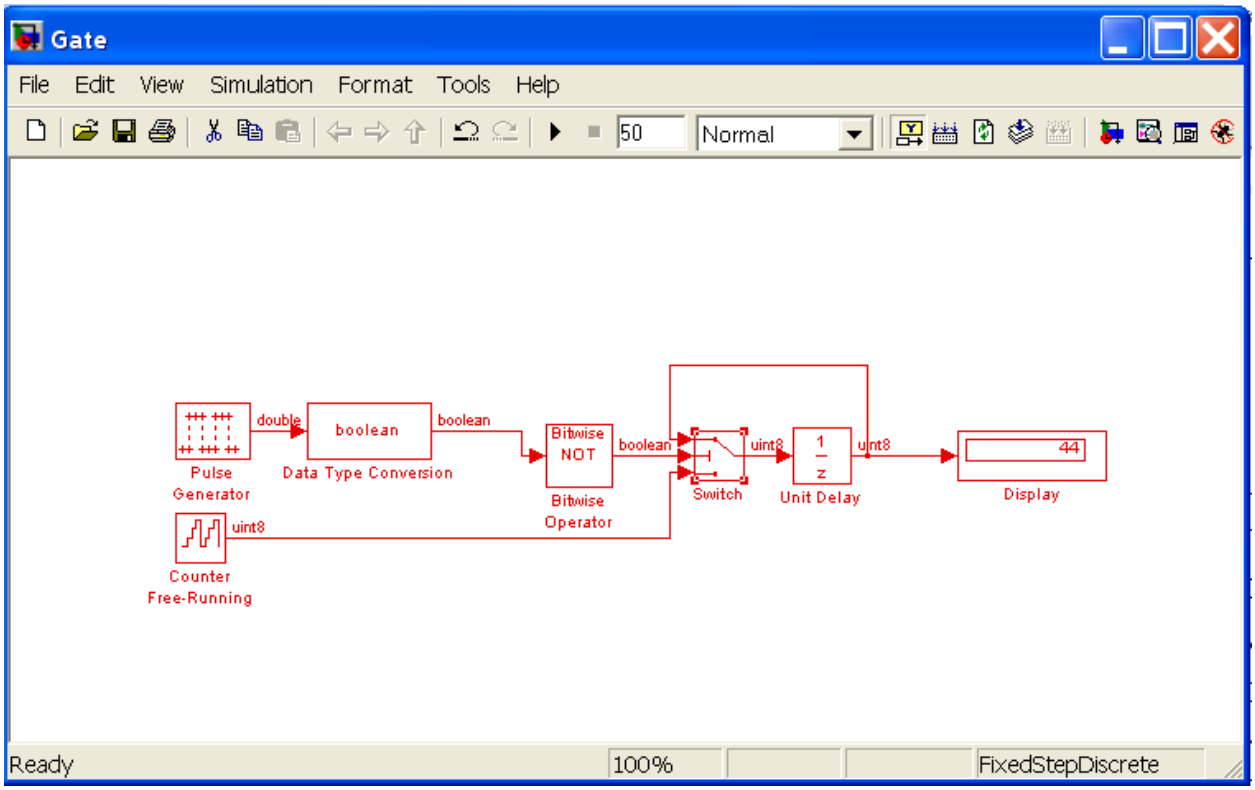


Рис. 16. Диаграмма модели управляемого вентиля

5. Выполните проверку работы модели, используя Simulink Debugger
6. Выполните генерацию VHDL-кода, предварительно объединив в подсистему все блоки модели, кроме Counter Free-Running, Pulse Generator, Data Type Conversion и Display.

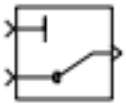
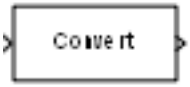
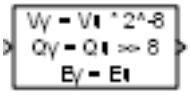
Приведем примеры решения еще двух задач. Первая из них – преобразование параллельного кода в последовательный. Вторая – обратно. Объединим обе задачи в одной модели так, что на выходе модели при ее правильной работе мы должны иметь повторение входного сигнала.

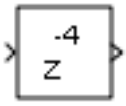
1. Создайте на диске директорию Parallel2Serial, запустите MatLab и укажите Parallel2Serial в качестве рабочей директории (Current Directory).
2. Запустите Simulink Library Browser и создайте новую модель. Сохраните ее под именем Prl2Srl.
3. Настройте параметры модели, запустив в Command Window среды MatLab скрипт

hdlsetup

В таблице ниже приведены блоки с кратким описанием их функциональности, которые необходимо поместить в диаграмму модели из Simulink Library Browser.

Символ и название	Категория в Simulink Library Browser	Функциональность
 Counter Free-Running	Sources	Источник байтового потока. В окне диалога Source Block Parameters в поле Number of bits установите 8. В поле Sample time установите 8.

Символ и название	Категория в Simulink Library Browser	Функциональность
 Extract Bits	Logic and Bit Operations	Выделяет из входного сигнала заданные разряды. В выпадающем списке Bits to extract окна диалога Function Block Parameters выберите Range of bits. В поле Bit indices установите [7 7]. В списке Output scaling mode выберите Treat bit field as an integer.
 Mux	Signal Routing	Объединяет входные сигналы в единый векторный сигнал. В поле Number of inputs окна диалога Function Block Parameters установите 8.
 Rate Transition	Signal Attributes	Согласовывает скорости работы входного и выходного портов. В поле Initial Condition окна диалога Function Block Parameters установите 0. В списке Output port sample time options выберите позицию Specify. В поле Output port sample time установите 1.
 Index Vector	Signal Routing	Выбирает из векторного сигнала элементы, индексы которых соответствую значению на управляющем порту, и отправляет их на выходной порт. Отметьте поле Use zero-based indexing окна диалога Function Block Parameters.
 Taped Delay	Discrete	Накапливает значения входного сигнала заданное количество тактов. Выходной сигнал представляет собой вектор из накопленных значений. В поле Sample Time окна диалога Function Block Parameters установите -1. В поле Number of Delays установите 8. В выпадающем списке Order output vector starting with выберите позицию Oldest
 Data Type Conversion	Signal Attributes	Осуществляет преобразование типа сигнала. В поле Output Data Type окна диалога Function Block Parameters установите uint8.
 Demux	Signal Routing	Разделяет векторный сигнал на отдельные скалярные значения. В поле Number of Outputs окна диалога Function Block Parameters установите 8.
 Shift Arithmetic	Logic and Bit Operations	Осуществляет битовый сдвиг (умножение на степень 2) входного сигнала. В поле Number of bits to shift right окна диалога Function Block Parameters установите -7.

Символ и название	Категория в Simulink Library Browser	Функциональность
 Bitwise Operator	Logic and Bit Operations	Выполняет побитовую операцию (какую-либо) над входными сигналами. В выпадающем списке Operator окна диалога Function Block Parameters выберите OR. В поле Number of input ports установите 8.
 Display	Sinks	Выводит на экран значения входного сигнала.
 Integer Delay	Discret	Запоминает значение входного сигнала на несколько тактов. На выходе значения входного сигнала появляются с задержкой в это количество тактов. В поле Number of delays окна диалога Function Block Parameters установите 2.

13. Размножьте необходимые блоки и соедините их сигналами так, чтобы диаграмма модели выглядела как на рис. 17.

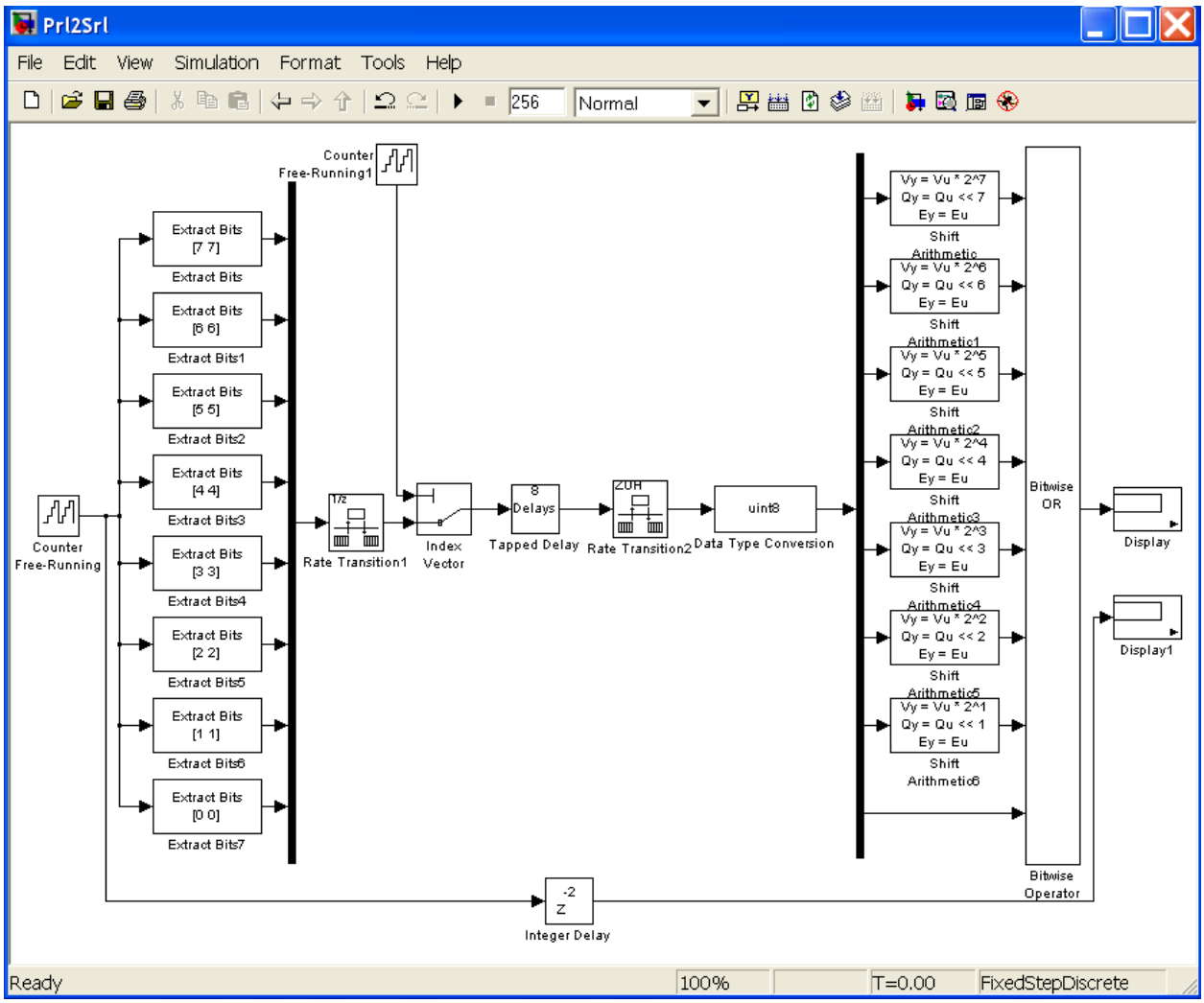


Рис. 17. Левая часть диаграммы (последний блок слева направо – Index Vector) реализует преобразование параллельного кода в последовательный. Правая часть – обратное преобразование. Блок Display1 отображает задержанную версию входного сигнала, генерируемого блоком Counter Free-Running

14. В окне диалога Source Block Parameters блока Counter Free-Running1 установите в поле Number of bits значение 3.
15. Установите параметры блоков Extract Bits … Extract Bits7 для извлечения битов входного сигнала с 7-го по 0-вой.
16. Установите значения параметров Number of bits to shift right блоков Shift Arithmetic … Shift Aritmetic 6 в -7 … -1.
17. Установите значение параметра Output port sample time блока Rate Transition2 равным 8.
18. Проверьте работу модели, используя Simulink Debugger.

Перед генерацией кода объедините блоки, относящиеся к преобразованию параллельного кода в последовательный, в одну подсистему, блоки, относящиеся к преобразованию последовательного кода в параллельный, – в другую. Блоки созданных подсистем объедините в подсистему верхнего уровня, которая и будет представлять собой моделируемое устройство. После объединения в окружение моделируемого цифрового устройства должны войти только блоки Counter Free-Running, Integer Delay, Display, Display1 (рис. 18, 19).

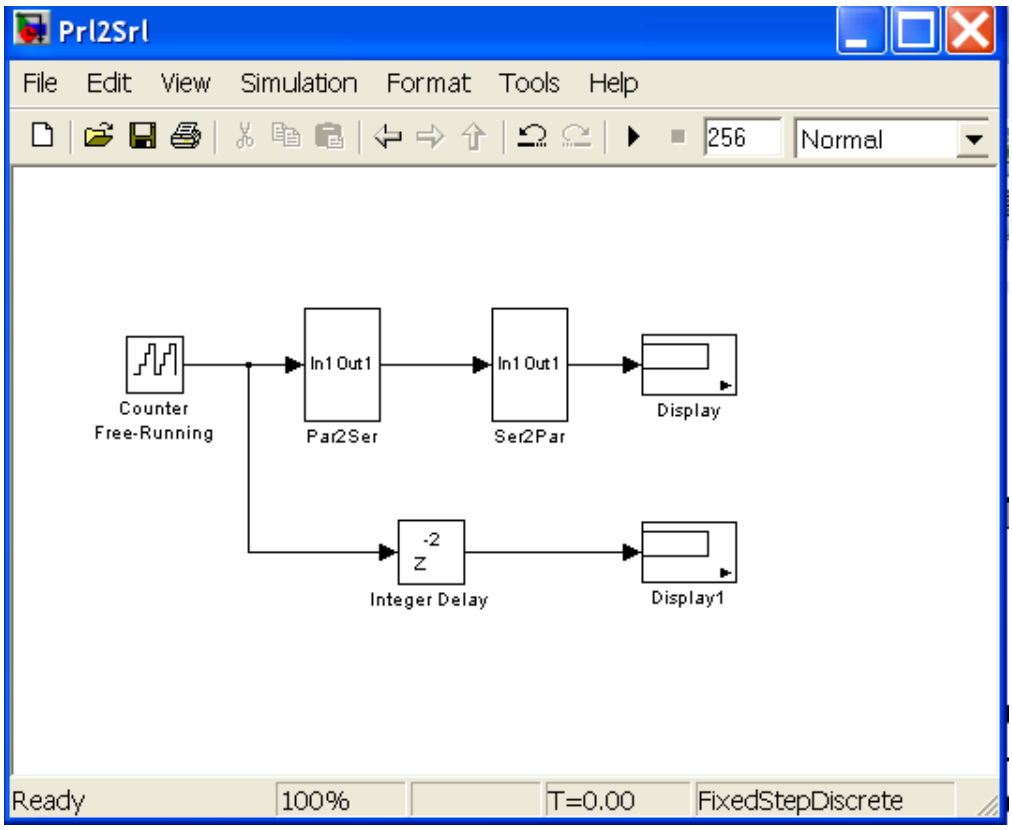


Рис. 18. Диаграмма модели после объединения блоков в подсистемы

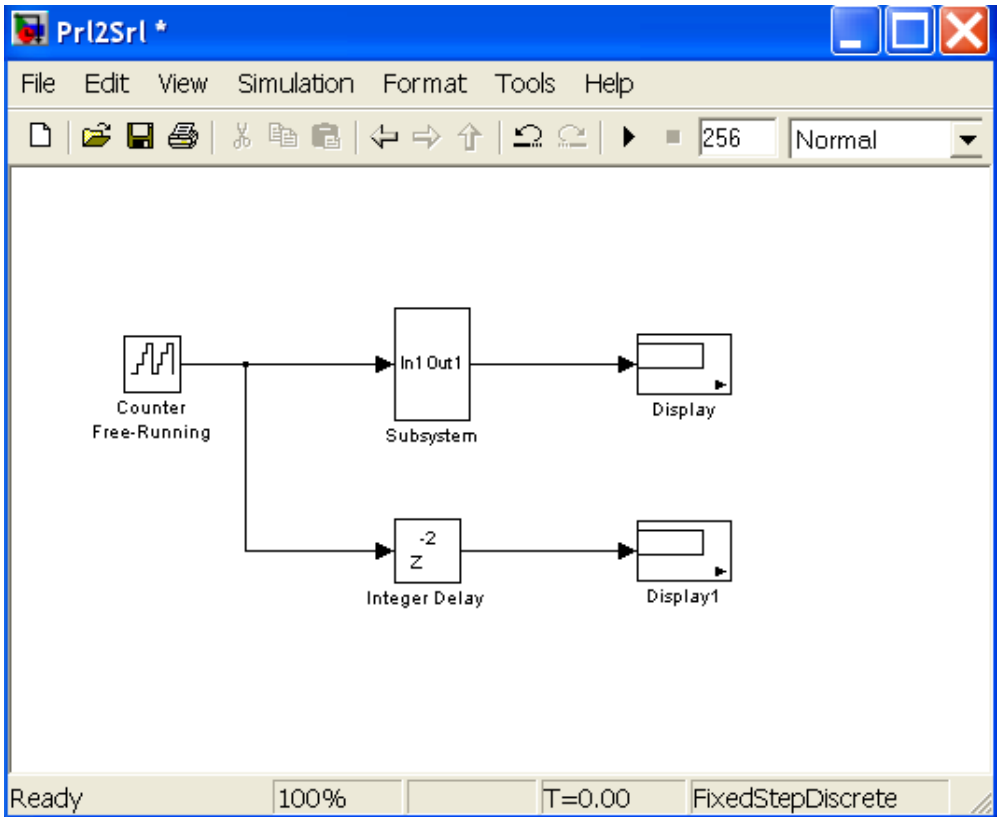


Рис. 19. Окончательный вид модели перед генерацией VHDL-кода

11. Выполните генерацию VHDL-кода модели и программы Testbench. Изучите сгенерированный код.

Блоки Embedded MatLab

Несмотря на то, что с Simulink поставляется богатый набор готовых блоков, которые можно использовать для моделирования цифровых устройств и совместимых с Simulink HDL Coder, в Simulink также предусмотрена возможность создания различных видов пользовательских (Custom) блоков. Функциональность таких блоков пользователь определяет сам. Из всех типов Custom блоков совместимы с HDL Coder только, так называемые, Embedded MatLab Function блоки. Функциональность таких блоков пользователь описывает, создавая программу на языке программирования Embedded MatLab.

Рассмотрим на примере создание модели с использованием Embedded MatLab Function блока. В модели реализуем счетчик, имеющий два двоичных управляющих сигнала: первый – rst, сбразывает счетчик в нулевое значение; второй – enable, разрешает или запрещает работу счетчика.

1. Создайте новую директорию, например Counter, и укажите ее в качестве Current Directory в программе MatLab.
2. Запустите Simulink и создайте новую модель. Сохраните ее под именем uint8_counter.
3. Выберите Embedded MatLab Function блок из раздела User-Defined Functions в Simulink Library Browser и поместите его в окно модели.
4. Откройте окно Embedded MatLab Editor двойным щелчком «мыши» по изображению блока на диаграмме модели.
5. Замените текст программы на следующий:

```
function y = counter(rst,enable)
persistent counter;

if isempty(counter)
    counter=uint8(0);
end

if rst
    counter=uint8(0);
elseif enable
    counter=counter+1;
end
y=counter;
```

6. Выберите блок Display из раздела Sinks в Simulink Library Browser и поместите его в окно модели.
7. Выберите блок Signal Builder из раздела Sources в Simulink Library Browser и поместите его в окно модели. Двойным щелчком «мыши» откройте окно Signal Builder. Используя меню открывшегося окна создайте два сигнала так, чтобы редактор сигналов выглядел как на рис. 20.

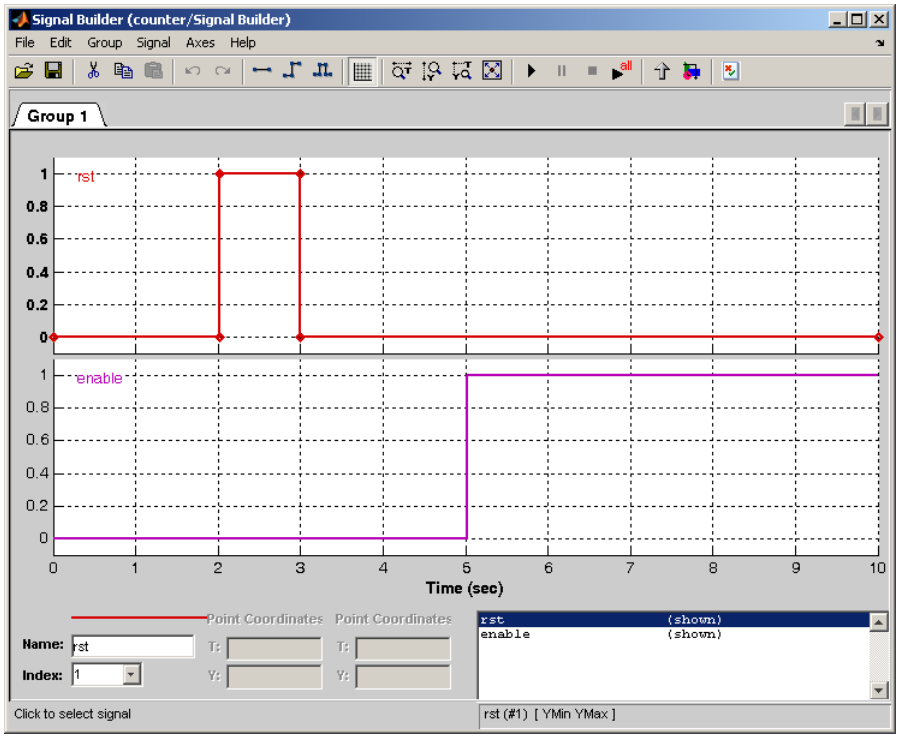


Рис. 20. Создайте два сигнала и откорректируйте положение импульсов на временной оси

8. Вызовите окно диалога Simulation Options, выбрав в меню File позицию Simulation Options. Из выпадающего списка Signal values after final time выберите Hold filnal value. В поле Sample time установите значение 1. Закройте окно диалога Signal Builder.
9. В окне Configuration Parameters модели в «дереве» Select выберите Solver. В поле Fixed step size установите значение 1.
10. Добавьте к диаграмме модели два блока Data Type Conversion из категории Signal Attributes в Simulink Library Browser. В настройках этих блоков установите в поле Output Data Type в значение Boolean.
11. Соедините блоки сигналами так, чтобы диаграмма модели выглядела как на рис. 21.

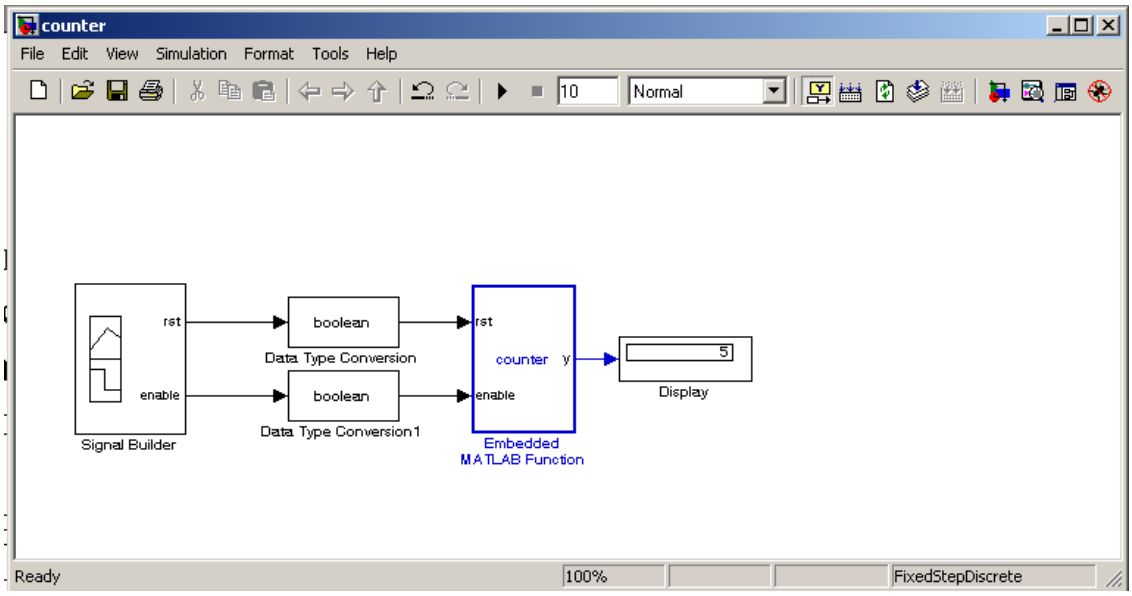


Рис. 21. Вид диаграммы модели, использующей Embedded MatLab function блок

12. Проверьте правильность работы модели, используя Simulink Debugger.

Разберемся с кодом, описывающим функциональность блока counter. Очевидно, что аргументы функции counter в коде программы отображаются во входные порты блока counter. Возвращаемое значение – y – является выходным портом. Отметим, что для задания нескольких выходных портов у Embedded MatLab function блока, имена портов перечисляются в квадратных скобках через запятую. Так, например, конструкция:

```
function [x,y,z] = counter(rst,enable)
```

описывает блок с тремя выходными портами с именами x, y и z.
Следующая конструкция:

```
persistent counter;
```

объявляет переменную counter как статическую переменную (persistent), значения которой сохраняются между вызовами функции. Необходимо заметить, что по умолчанию все переменные в Embedded MatLab function являются локальными и их значения не сохраняются между вызовами функции (вызов Embedded MatLab function происходит на каждом временном шаге работы модели при симуляции). Указанная выше конструкция является просто объявлением переменной. Захват памяти и инициализация переменной counter производится следующими операторами:

```
if isempty(counter)
    counter=uint8(0);
end
```

Здесь функция isempty возвращает значение TRUE, если переменная counter еще не существует. Таким образом, инициализация и захват памяти под статическую переменную производится только один раз при инициализации модели.

Еще одно важное замечание. По умолчанию, в языке MatLab все переменные считаются переменными типа double. Все остальные типы данных необходимо указывать явно при присвоении переменной значения. Остальные конструкции, используемые в функции counter, думается, не требую каких-либо пояснений.

И еще одно замечание. Вместе с Simulink HDL Coder поставляется богатая библиотека примеров Embedded MatLab function блоков. Доступ к этому набору готовых блоков с редактируемой функциональностью можно получить, набрав в Command Window программной среды MatLab команду

eml_hdl_design_patterns

Сигналы произвольной разрядности. Арифметика с фиксированной точкой

Возможность работы с сигналами произвольной разрядности является насущной необходимостью при моделировании цифровых устройств. Очевидно, что встроенные типы данных системы MatLab/Simulink (double, boolean, uint8, uint16, uint32, int8, int16, int32) не предоставляют такой возможности. С другой стороны, вместе с MatLab/Simulink поставляется Fixed Point Toolbox и Simulink Fixed Point, где реализован мощный аппарат, позволяющий создавать собственные типы данных и, в том числе, данные произвольной разрядности.

Основным инструментом для создания переменных и сигналов произвольной разрядности является функция fi(), возвращающая, т.н., fi-объект. Рассмотрим основные свойства и способы создания fi-объектов, наиболее часто используемые в Simulink-диаграммах для моделирования цифровых устройств.

Синтаксис вызова функции fi() выглядит следующим образом:

a=fi(v,s,w,f,F),

где
v – значение создаваемого числа произвольной разрядности;
s – определяет является ли создаваемое число **a** числом со знаком; если **s** равно 1, то **a** – число со знаком (может принимать как положительные, так и отрицательные значения); в

противном случае, когда **s** равно 0, **a** является беззнаковым числом (принимает только положительные значения);
w – количество разрядов в создаваемом числе **a** (в моделях, совместимых с Simulink HDL Coder, **w** не должно превышать значения 32);
f – задает количество знаков после запятой для чисел с фиксированной точкой; если **f** равно 0, то **a** является целым числом;
F – задает правила выполнения арифметических операций над создаваемым числом; **F** является, т.н. fimath-объектом, который можно создать с помощью функции fimath().

Рассмотрим основные свойства и синтаксис вызова функции fimath():

F=fimath(… , 'PropertyName', 'PropertyValue', …),

где PropertyName – имя задаваемого свойства объекта F; PropertyValue – его значение. Ниже перечислены некоторые свойства fimath-объекта и их возможные значения.

RoundMode. Задает правила округления при выполнении арифметических операций с числами с фиксированной точкой. Возможные значения этого свойства:

ceil – округление в сторону плюс бесконечности (рис. 22);

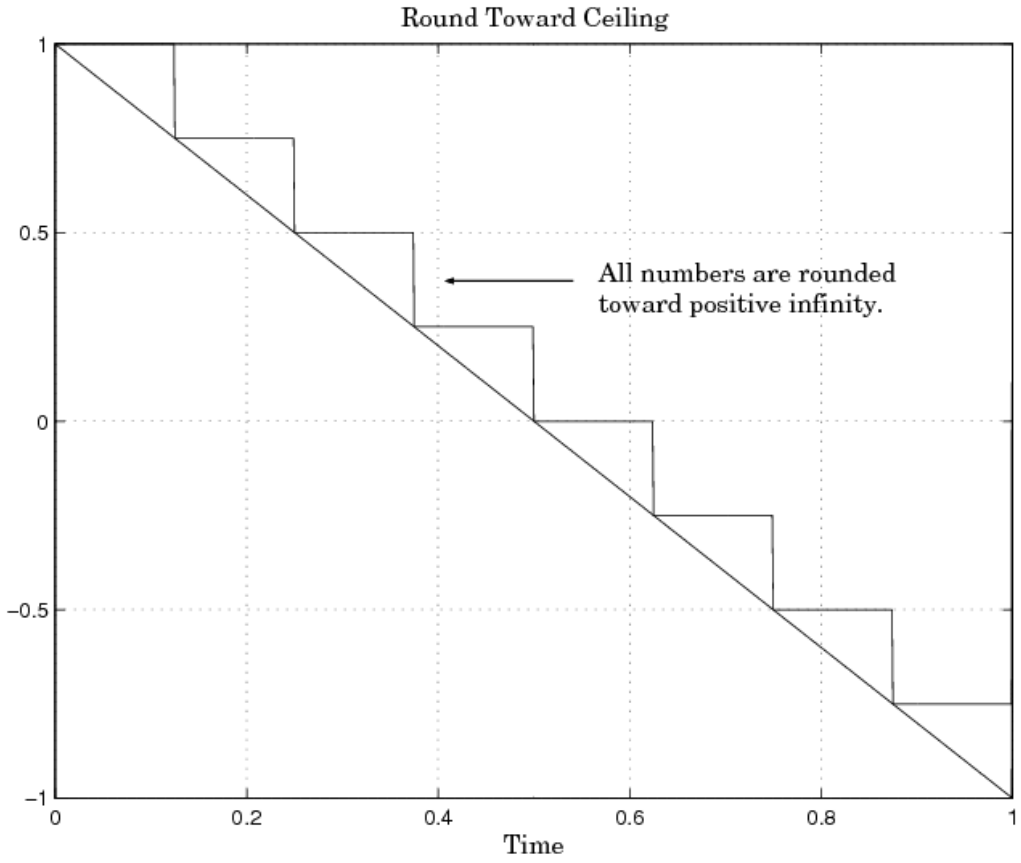


Рис. 22. Округление ceil

fix – округление в сторону 0 (рис. 23);

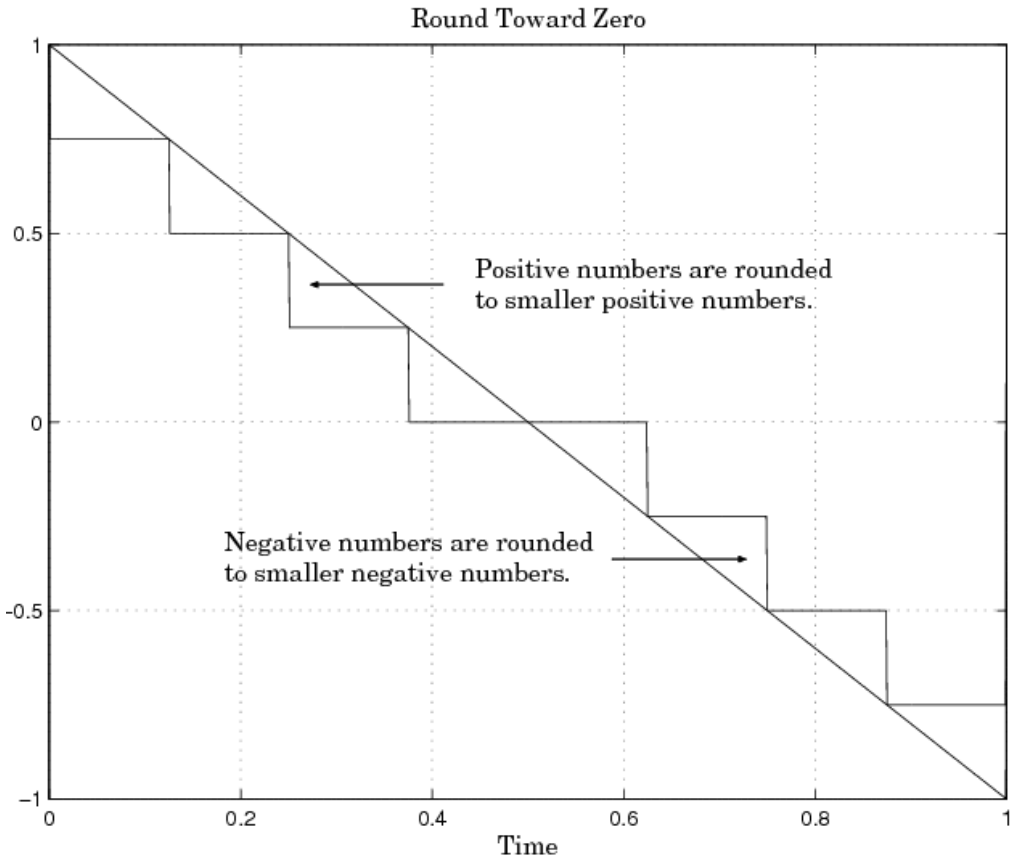


Рис. 23. Округление fix

floor – округление в сторону минус бесконечности (рис.24);

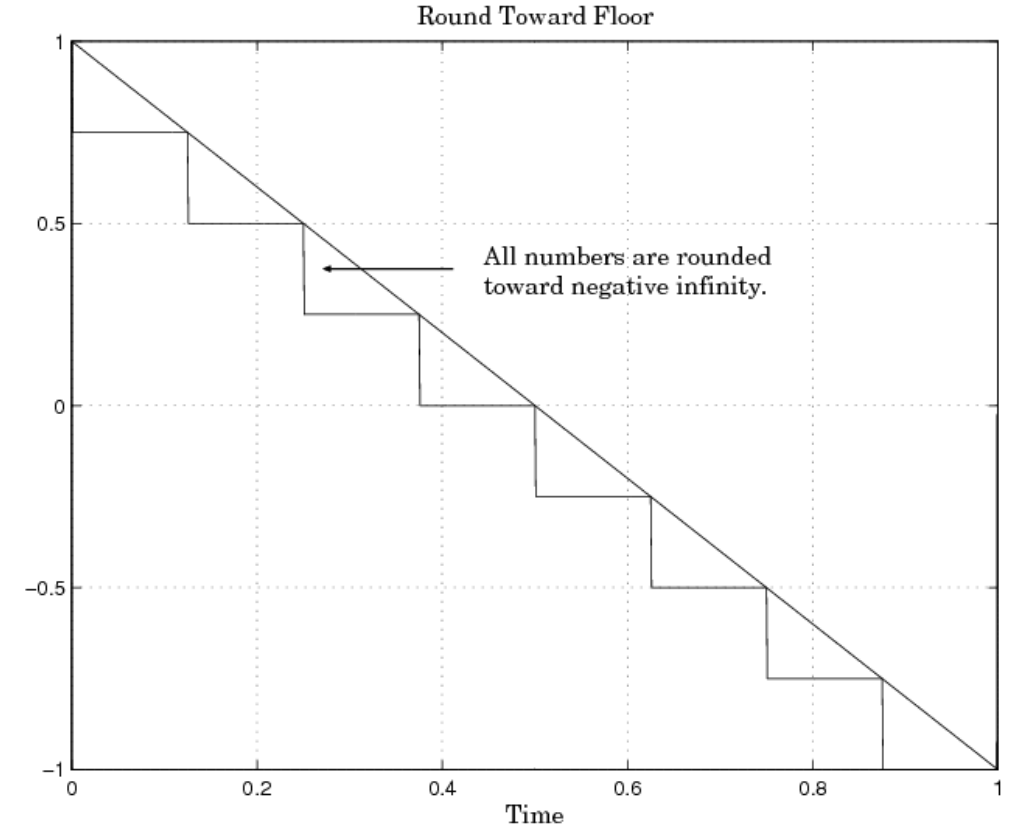


Рис. 24. Округление floor

nearest – округление с минимальной ошибкой (рис.25).

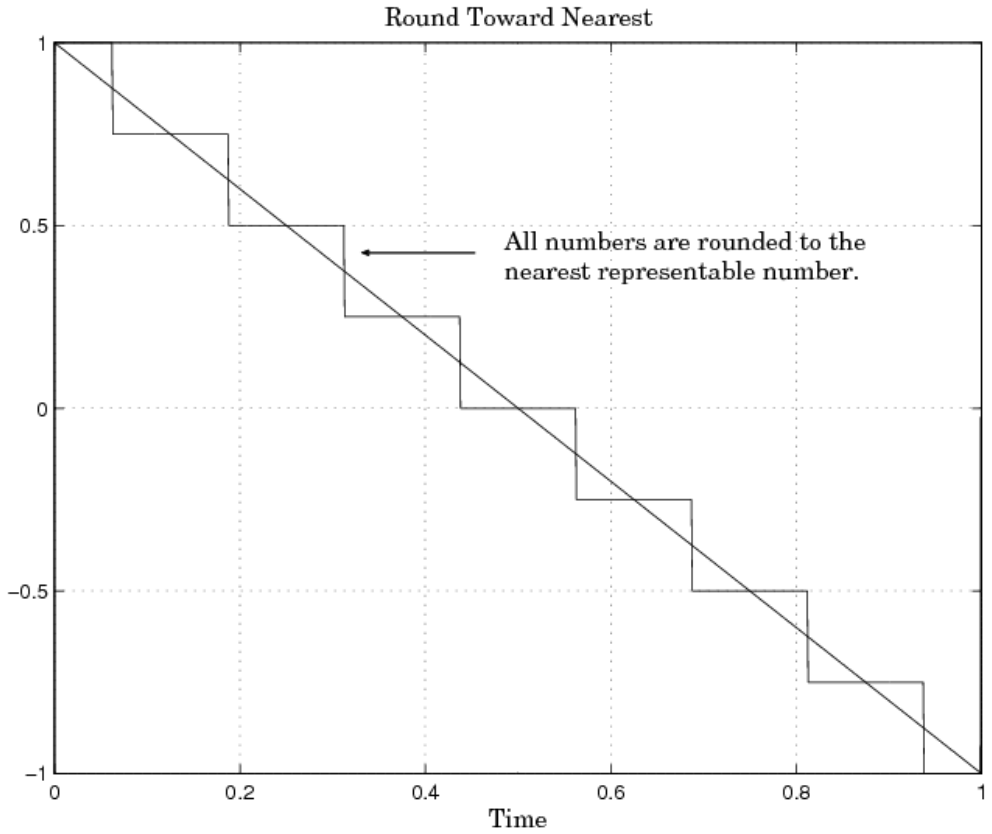


Рис. 25. Округление nearest

OverflowMode. Задает операцию, выполняемую при переполнении (когда результат выполнения арифметической операции превышает максимально возможное значение числа с фиксированной точкой). Возможные значения этого свойства:

saturate – при переполнении число с фиксированной точкой имеет максимально возможное значение;

wrap – при переполнении результат вычисляется по модулю максимально возможного значения (например, если к 2-х разрядному числу 3 прибавить 1, то результат будет равен 1)

ProductMode. Определяет правило, по которому определяется тип данных результата умножения двух чисел с фиксированной точкой. Возможные значения этого свойства (при описании возможных значений свойства ProductMode используются следующие обозначения. W_a , W_b – количество разрядов в сомножителях; W_p – количество разрядов результата произведения; F_a , F_b – количество разрядов после запятой у сомножителей; F_p – количество разрядов после запятой у произведения):

FullPrecision – сохранение точности результата.

$$W_p = W_a + W_b;$$

$$F_p = F_a + F_b.$$

Если значение W_p превышает значение MaxProductWordLength, то генерируется ошибка.

KeepLSB – сохранение значения младшего значащего бита. Величина W_p задается значением свойства ProductWordLength.

$$F_p = F_a + F_b$$

Точность результата произведения чисел сохраняется. Однако, возможно переполнение.

KeepMSB – сохранение значения старшего бита. Величина W_p задается значением свойства ProductWordLength.

$$F_p = W_p - (W_a + W_b) + (F_a - F_b)$$

Возможна потеря точности результата произведения.

SpecifyPrecision – заданная точность. Величина W_p задается значением свойства ProductWordLength. Величина F_p задается значением свойства ProductFractionLength.

MaxProductWordLength. Определяет максимальное число разрядов результата перемножения двух чисел с фиксированной точкой. В моделях, совместимых с Simulink HDL Coder, разрядность всех чисел не должна превышать 32.

SumMode. Определяет правило, по которому определяется тип данных результата сложения двух чисел с фиксированной точкой. Возможные значения этого свойства (при описании возможных значений свойства SumMode используются следующие обозначения. W_a , W_b – количество разрядов слагаемых; W_s – количество разрядов результата; F_a , F_b – количество разрядов после запятой у слагаемых; F_s – количество разрядов после запятой у суммы):

FullPrecision – сохранение точности результата.

$$W_s = integer\_length + F_s;$$

$$integer\_length = \max(W_a - F_a, W_b - F_b) + 1;$$

$$F_s = F_a + F_b.$$

Если значение W_s превышает значение MaxSumWordLength, то генерируется ошибка.

KeepLSB – сохранение значения младшего значащего бита. Величина W_s задается значением свойства SumWordLength.

$$F_s = \max(F_a, F_b)$$

Точность результата произведения чисел сохраняется. Однако, возможно переполнение.

KeepMSB – сохранение значения старшего бита. Величина W_s задается значением свойства SumWordLength.

$$F_s = W_s - integer\_length,$$

$$integer\_length = \max(W_a - F_a, W_b - F_b) + 1$$

Возможна потеря точности результата суммы.

SpecifyPrecision – заданная точность. Величина W_s задается значением свойства SumWordLength. Величина F_s задается значением свойства SumFractionLength.

MaxSumWordLength. Определяет максимальное число разрядов результата сложения двух чисел с фиксированной точкой. В моделях, совместимых с Simulink HDL Coder, разрядность всех чисел не должна превышать 32.

CastBeforeSum. Может принимать два значения: TRUE и FALSE. Если TRUE, перед выполнением операции сложения операнды приводятся к типу данных результата.

Рассмотрим пример. Наберите в Command Window программной среды MatLab команду:

F=fimath('RoundMode','nearest','OverflowMode','wrap',...

'ProductMode','SpecifyPrecision','ProductWordLength',32,...

'ProductFractionLength',15,'MaxProductWordLength',32,...

'SumMode','SpecifyPrecision','SumWordLength',32,...

'SumFractionLength',15,'MaxSumWordLength',32,...

'CastBeforeSum',true)

После нажатия клавиши Enter на экране появится результат выполнения команды – отобразится созданный объект **F**:

```
          RoundMode:  nearest
       OverflowMode:  wrap
        ProductMode:  SpecifyPrecision
  ProductWordLength:  32
ProductFractionLength:  15
            SumMode:  SpecifyPrecision
      SumWordLength:  32
   SumFractionLength:  15
      CastBeforeSum:  true
```

С созданным объектом F создадим несколько чисел. Для этого наберите в Command Window программной среды MatLab команду:

a=fi(3.14159265,1,32,15,F)

Результат отобразится на экране в виде:

```
3.1416
DataTypeMode: Fixed-point: binary point scaling
          Signed:  true
      WordLength:  32
  FractionLength:  15

          RoundMode:  nearest
       OverflowMode:  wrap
        ProductMode:  SpecifyPrecision
  ProductWordLength:  32
ProductFractionLength:  15
            SumMode:  SpecifyPrecision
      SumWordLength:  32
   SumFractionLength:  15
      CastBeforeSum:  true
```

Наберите команду

b=2*a

Результат:

```
6.2832
        DataTypeMode:  Fixed-point: binary point scaling
              Signed:  true
          WordLength:  32
      FractionLength:  15

          RoundMode:  nearest
       OverflowMode:  wrap
        ProductMode:  SpecifyPrecision
  ProductWordLength:  32
ProductFractionLength:  15
            SumMode:  SpecifyPrecision
```

SumWordLength:	32
SumFractionLength:	15
CastBeforeSum:	true

Наберите команду

C=a+b

Результат:
9.4248

DataTypeMode:	Fixed-point: binary point scaling
Signed:	true
WordLength:	32
FractionLength:	15
RoundMode:	nearest
OverflowMode:	wrap
ProductMode:	SpecifyPrecision
ProductWordLength:	32
ProductFractionLength:	15
SumMode:	SpecifyPrecision
SumWordLength:	32
SumFractionLength:	15
CastBeforeSum:	true

Перебирая арифметические операции попробуйте их различные комбинации над объектами **a**,**b** и **c**. Анализируйте точность результата после выполнения каждого арифметического действия (Для деления чисел с фиксированной точностью используйте функцию divide()).

Числовые fi-объекты можно использовать в моделях цифровых устройств в Embedded MatLab function блоках. Рассмотрим создание таких объектов на примере 4-х разрядного счетчика.

1. Создайте директорию Counter на диске и укажите ее в качестве рабочей директории программной среды MatLab.
2. Запустите Simulink Library Browser и создайте окно новой модели. Сохраните ее под именем Counter4.
3. Настройте параметры модели для генерации HDL-кода выполнив скрипт hdlsetup в окне Command Window среды MatLab.
4. Поместите в окно модели блок Embedded MatLab function из категории User-Defined Functions в Simulink Library Browser.
5. Запустите редактор Embedded MatLab Editor, дважды щелкнув левой кнопкой «мыши» по новому блоку.
6. Наберите следующий код.

```
function count = Counter

result_fm = fimath('OverflowMode', 'wrap');

persistent count_reg;
if isempty(count_reg)
    count_reg = fi(0, 0, 4, 0, result_fm);
end

count = count_reg;
```

```
count_reg = fi(count_reg + 1, 0, 4, 0, result_fm);
```

7. Добавьте в модель блок Display из категории Sinks в Simulink Library Browser.
8. Скомпилируйте модель и запустите ее в пошаговом режиме, используя Simulink Debugger.
9. Проанализируйте код Embedded MatLab function блока, использующего fi-объекты в вычислениях.

Можно указать также fimath-свойства входных сигналов (то есть указать атрибуты сигналов, которые являются допустимыми на входе данного блока). Для этого необходимо отрыть Embedded MatLab Editor для кода данного блока (дважды щелкнув по нему левой кнопкой «мыши») и в панели инструментов открывшегося редактора нажать на кнопку Edit Data/Ports (рис. 26).

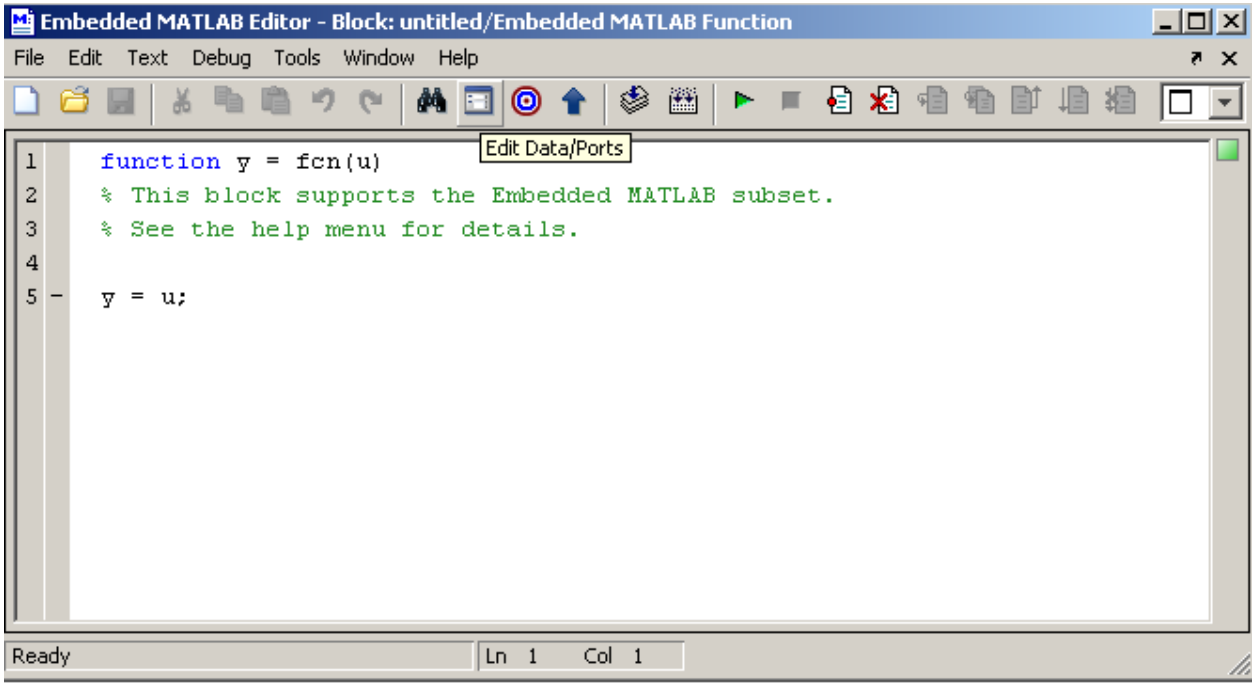


Рис. 26. Для задания fimath-свойств нажмите на кнопку Edit Data/Ports

В открывшемся окне Ports and Data Manager (рис. 27) укажите необходимые fimath-свойства.

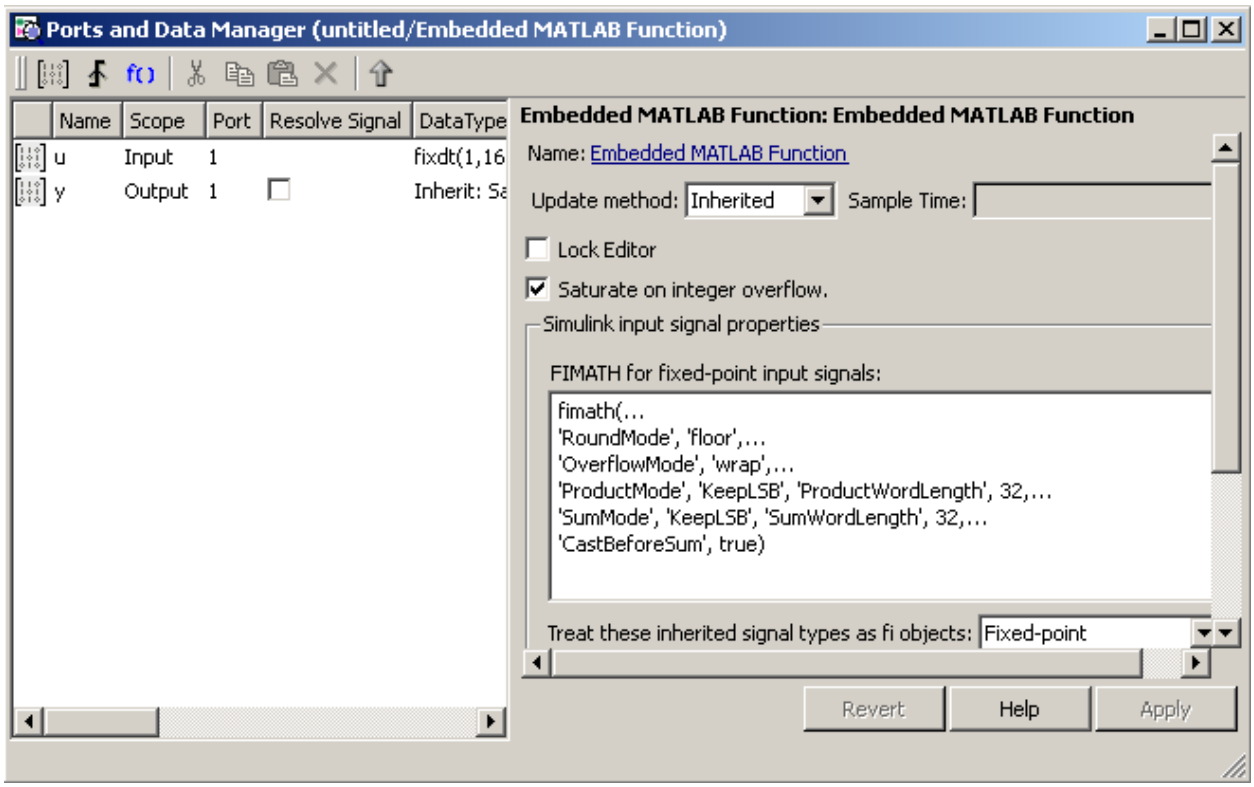


Рис. 27. В окне FIMATH for fixed-point input signals можно указать ожидаемые (допустимые) свойства входных сигналов

fi-объекты могут быть созданы и с помощью стандартных блоков из Simulink Library browser. Так, например, рассмотрим блок Constant из категории Sources в Simulink Library browser. На странице параметров блока, имеющей название Signal Attributes, выпадающий список Output Data Type позволяет установить тип выходного порта на значение fixdt(1,16,0) (рис. 28).

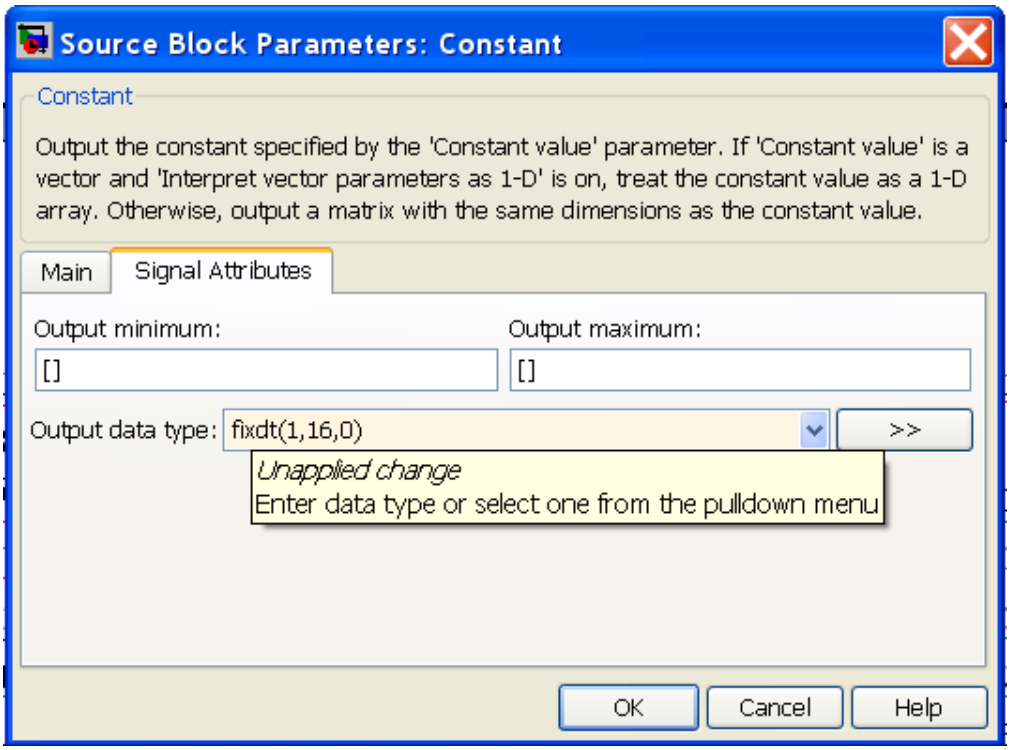


Рис. 28. Задание типа fi-объекта а качестве типа выходного порта блока Constant

Таким образом, тип выходного сигнала для блока устанавливается в 16-ти разрядное целое число со знаком (аргументы функции fixdt имеют то же значение, что и у функции fi). Меняя значения аргументов fixdt легко получить числа произвольной разрядности (до 32) с произвольным положением фиксированнной точки.

Рассмотрим еще один полезный пример. Реализуем модель FIFO-буфера длины width, которая при имплементации VHDL-кода в FPGA-микросхему будет использовать блочную память. При реализации модели будем использовать полезную заготовку, поставляемую вместе с Simulink HDL Coder.

1. Создайте директорию FIFO на диске и укажите ее в качестве рабочей директории программной среды MatLab.
2. Запустите Simulink Library Browser и создайте окно новой модели. Сохраните ее под именем FIFO.
3. Настройте параметры модели для генерации HDL-кода выполнив скрипт hdlsetup в окне Command Window среды MatLab.
4. Поместите в окно модели блок Embedded MatLab function из категории User-Defined Functions в Simulink Library Browser.
5. Запустите редактор Embedded MatLab Editor, дважды щелкнув левой кнопкой «мыши» по новому блоку.
6. Наберите следующий код.

```
function [wr_addr,rd_addr,d_val] = addr_gen(width,reset,wr_en)
persistent w_a;
persistent r_a;

persistent state;
F=hdlfimath;
if isempty(w_a)
    w_a=fi(1023,0,10,0,F);
end

if isempty(r_a)
    r_a=fi(1023,0,10,0,F);
end

if isempty(state)
    state=false;
end

if reset
    w_a=fi(0,0,10,0,F);
    r_a=fi(0,0,10,0,F);
    state=false;
end

if wr_en
    w_a=fi(w_a+1,0,10,0,F);
    if w_a>fi(width,0,10,0,F)
        w_a=fi(0,0,10,0,F);
        if state==false
            state=true;
        end
    end
    if state
        r_a=fi(r_a+1,0,10,0,F);
```

```
    end
    if r_a>fi(width,0,10,0,F)
        r_a=fi(0,0,10,0,F);
    end
end

wr_addr=w_a;
rd_addr=r_a;
d_val=state;
```

7. В окне Command Window программной среды MatLab наберите команду

hdldemolib

8. Поместите на диаграмму модели блок Dual Port RAM из открывшегося окна Library:hdldemolib (Это и есть та полезная заготовка, о которой упоминалось выше).
9. Поместите на диаграмму модели блоки из Simulink Library Browser и соедините их так, чтобы диаграммы модели выглядела как на рис. 29.

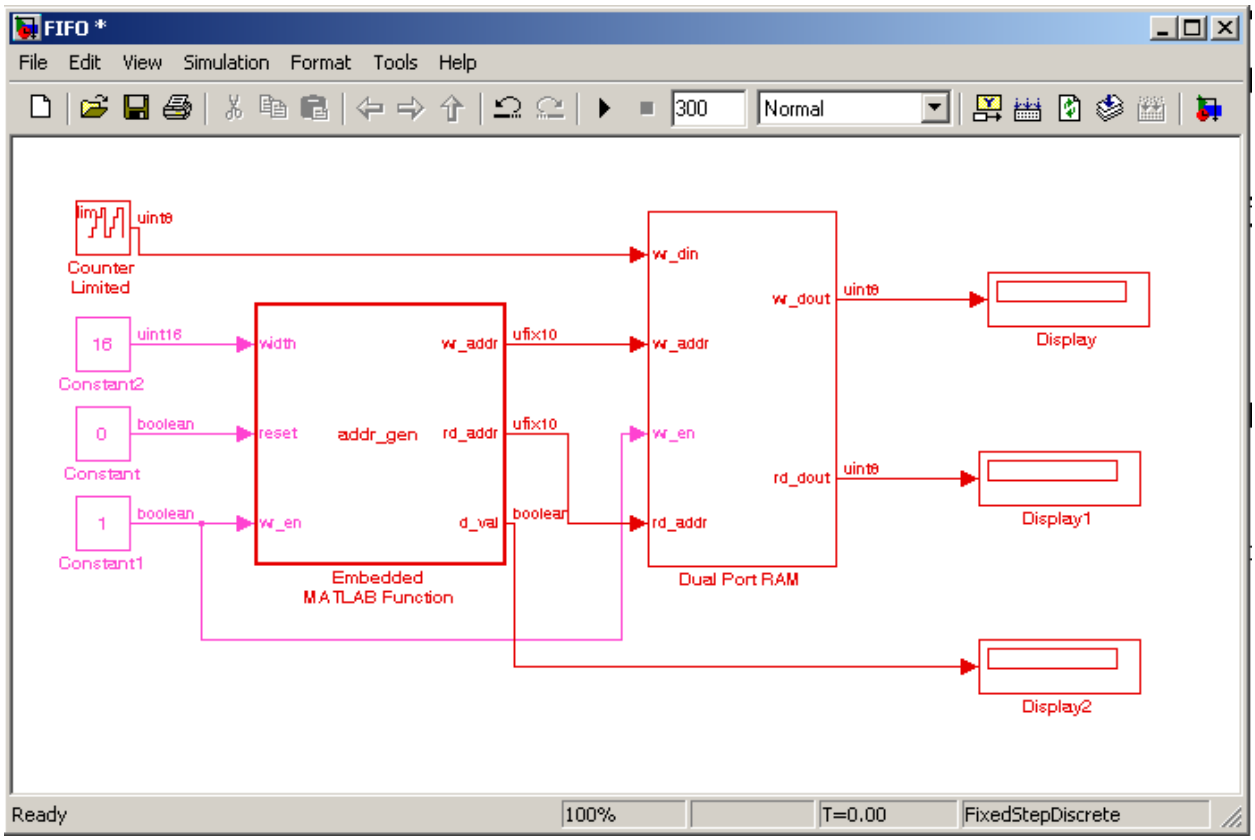


Рис. 29. Вид модели FIFO-буфера

10. Настройте параметры блоков Constant, Constant1, Constant2. Значения в поле Constant value установите в 0, 1, 16 соответственно. На закладке Signal Attributes у первых двух блоков выберите тип boolean, у последнего – uint16.
11. Настройте параметры блока Counter Limited (этот блок в модели имитирует поток данных, поступающих на вход FIFO-буфера). В поле Upper Limit установите значение 255.
12. Настройте параметры блока Dual Port RAM, установив в поле Address Port width значение 10.
13. Скомпилируйте и проверьте работу модели, используя Simulink Debugger.
14. Объедините блоки addr_gen и Dual Port RAM в подсистему. Назовите входные и выходные порты подсистемы так, как это показано на рис. 30.

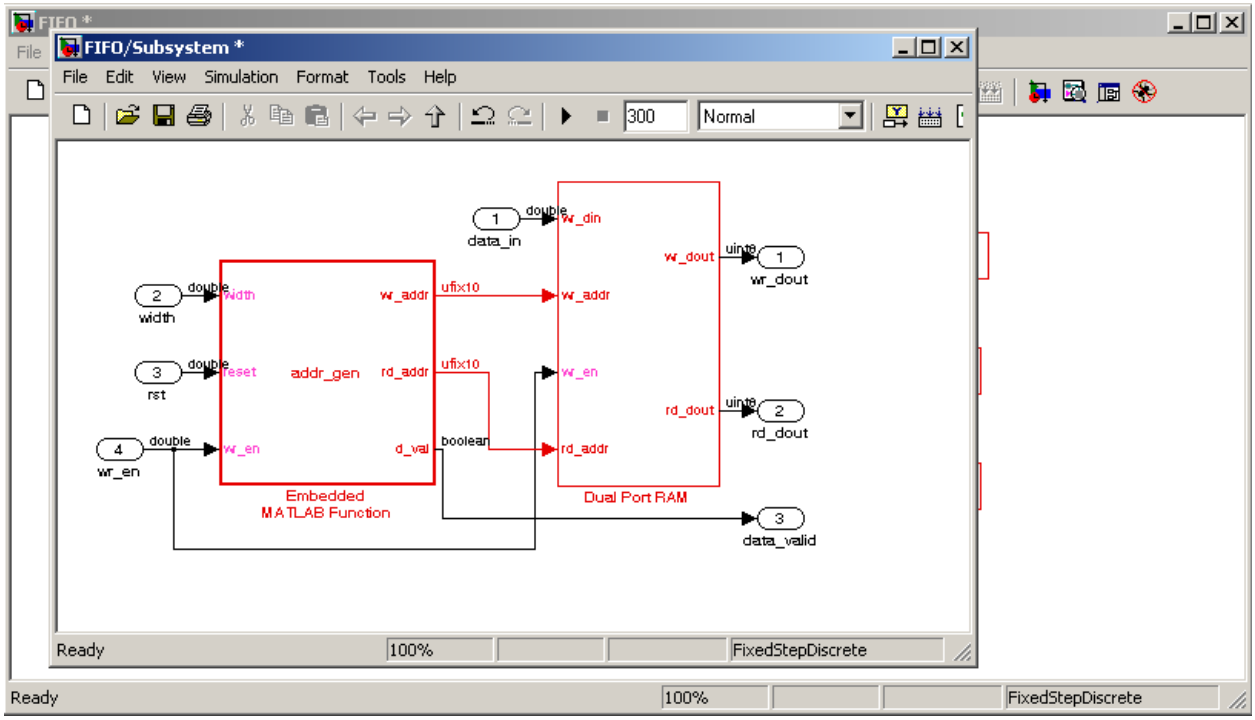


Рис. 30. Переопределите имена входных и выходных портов

15. Переопределите имя созданной подсистемы на FIFO_buf.
16. В окне Configuration Parameters модели в настройках для HDL Coder в поле Generate HDL for укажите FIFO/FIFO_buf.
17. В окне Command Window среды MatLab запустите скрипт

hdlnewcontrolfile

18. Сохраните открывшийся control file под именем FIFOControlFile.m
19. Откройте окно подсистемы FIFO_buf и выберите блок Dual Port RAM, щелкнув по нему кнопкой «мыши».
20. В окне Command Window среды MatLab запустите на выполнение скрипт

hdlnewforeach

В результате работы срипта в окне Command Window появится текст

```
c.forEach('FIFO/FIFO_buf/Dual Port RAM',...
 'hdldemolib/Dual Port RAM', {},...
 'hdldefaults.RamBlockDualHDLInstantiation', {});
```

21. Скопируйте этот текст в конец файла FIFOControlFile.m.
22. В окне Configuration Parameters модели в настройках для HDL Coder нажмите на кнопку Load и укажите FIFOControlFile.m в качестве control file для модели. В результате окно Configuration Parameters должно выглядеть как на рис. 31.

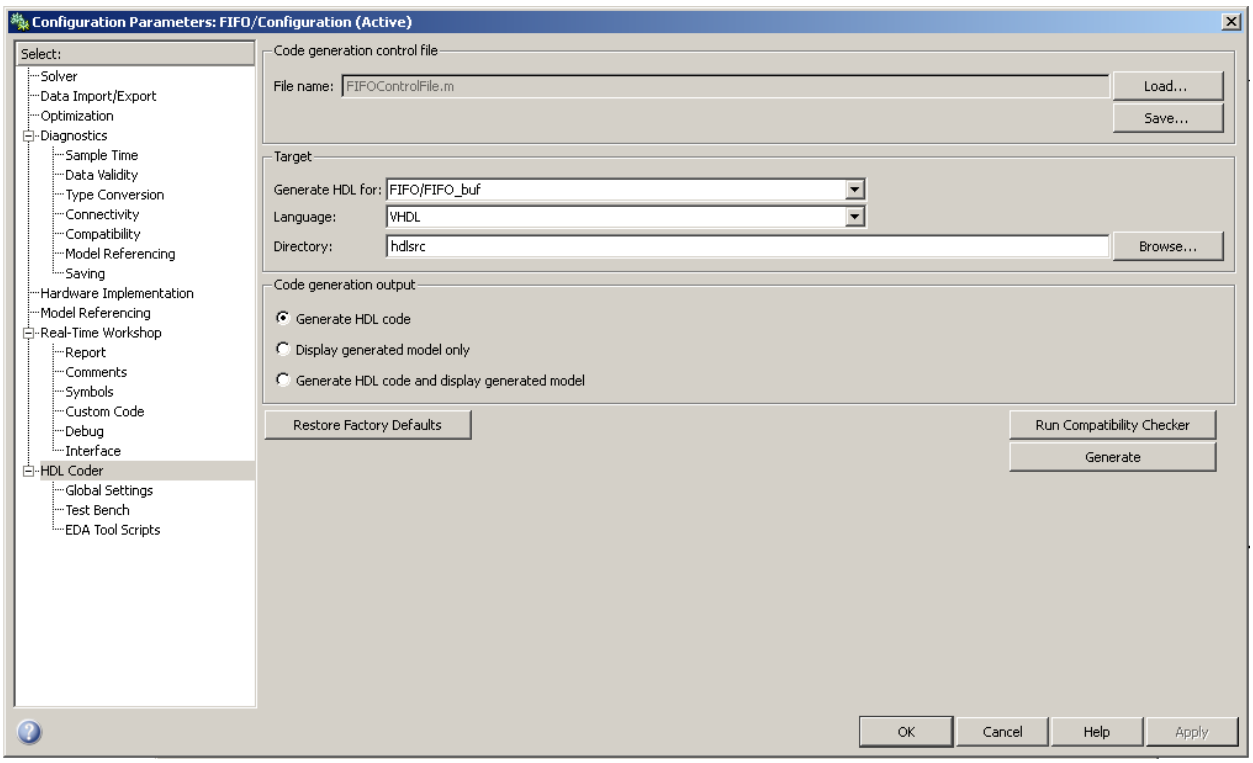


Рис. 31. Настройки HDL Coder готовы к генерации VHDL-кода модели

23. Выполните проверку совместимости модели с HDL Coder, нажав на кнопку Run Compatibility Checker.
24. Выполните генерацию VHDL-кода, нажав на кнопку Generate.
25. Переключитесь на закладку Test Bench и выполните генерацию тестирующей программы, нажав на кнопку Generate Test Bench.
26. Проанализируйте сгенерированный код.