Графическое программирование для DSP — без C, без ассемблера

Шекар ШАРАД (Shekhar SHARAD)
Перевод: Иван ГРИГОРЬЕВ

В настоящее время системы искусственного интеллекта развиваются по экспоненциальной зависимости. От машин до игровых приставок, от микроволновых печей до паровых утюгов — в каждом объекте имеется какая-то доля интеллекта.

Введение

Сейчас разработка встраиваемых систем уже не является узко специализированной областью: в сообщество разработчиков встраиваемых систем входят специалисты в различных областях. Одни из основных компонентов встраиваемых систем — это сигнальные процессоры (ЦСП), вследствие присущих им производительности, малых размеров, низкой стоимости и небольшого энергопотребления. С увеличением функциональности и сложности создаваемых систем и в связи с разным уровнем подготовки разработчиков появляется необходимость в использовании высокоуровневой среды, позволяющей специалистам в отдельно взятой области быстрее внедрять алгоритмы цифровой обработки сигналов и тем самым сократить временные затраты на макетирование промежуточных этапов и рутинное изучение специализированных средств разработки. Парадигмы графического программирования обращены именно на решение этой проблемы. В этой статье мы расскажем, как технологии графического программирования позволяют создавать приложения для целевых плат цифровой обработки сигналов и проводить весь цикл разработки, макетирования и внедрения в одной среде.

Графическое программирование

Использование блок-диаграмм — довольно распространенный метод представления функциональной части готового и будущего проекта. Графическое программирование основано на концепции «потоков данных», которая математически описывается ориентированным графом, в котором узлы предназначены для вычисления, а ветви являются потоками данных. В узлы могут входить подсистемы, содержащие последовательности узлов и ветвей, или алгоритмы, представленные на базовых языках, вроде C или ассемблера. Можно сказать, что применение языков графического программирования естественно для создания встраиваемых приложений и программирования сигнальных процессоров, результатом чего является широкое использование данной технологии в исследованиях и разработках.

Существует несколько средств разработки, позволяющих совмещать технологии визуального и текстового программирования, например Signal [1], Lustre [2] и Silage [3]. Полностью графическое программное окружение Ptolemy [4], разработанное Калифорнийским Университетом Беркли, а также LabVIEW компании National Instruments используются в самых разнообразных областях науки и техники. LabVIEW [5] можно применять для программирования широкого круга систем, от настольных ПК и переносных компьютеров до систем реального времени, плат на базе ПЛИС, 32-битных программируемых контроллеров и сигнальных процессоров. Одна из сильных сторон языков графического программирования — то, что пользователь имеет возможность переносить одно и то же приложение на множество объектов, с незначительной модификацией или без изменения исходного кода. Еще одно важное достоинство графического программирования — то, что у разработчика нет необходимости взаимодействовать с низкоуровневым кодом, который создается для соответствующего объекта. Фактически разработчик всегда работает на более высоком уровне абстракции (в графическом интерфейсе), что позволяет ему упростить создание сложных приложений, которые затем могут быть загружены на целевую платформу, в том числе сигнальный процессор.

Графическое программирование как базовый инструмент для инженеров

Как было отмечено ранее, цифровая обработка сигналов (ЦОС) и сигнальные процессоры (ЦСП) сейчас применяются практически везде, причем множество приложений создают специалисты, которые не являются экспертами в области ЦОС. Основная задача для них — это оптимальное использование готовых алгоритмов ЦОС в своих проектах. Традиционные методы программирования ЦСП предъявляют высокие требования к квалификации, что усложняет их использование, особенно для инженеров, не являющихся специалистами в области разработки решений по ЦОС. Графические языки программирования снимают эту проблему, предоставляя простой, легкий в использовании интерфейс для быстрой разработки, макетирования и внедрения готовых систем. На рис. 1 сравниваются шаги, которые совершаются при программировании ЦСП традиционным спо-

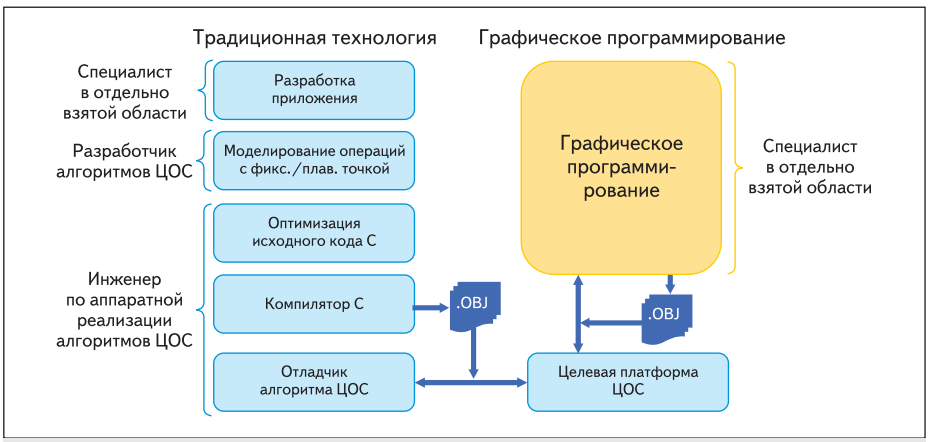

Рис. 1. Сравнение шагов, совершаемых при традиционном подходе к программированию ЦСП и с использованием методов графического программирования

собом и при помощи технологии графического программирования.

Как показано на рис. 1, используя традиционные технологии, разработчик должен уверенно разбираться в алгоритмах ЦОС, либо уметь взаимодействовать со специалистами в этой области. Графическое программирование позволяет ему сосредоточиться на разработке своего проекта в целом, что снижает время на разработку и выход готового изделия на рынок. Средства графического программирования дают возможность большему числу инженеров внедрять алгоритмы ЦОС в свои приложения, прежде всего за счет удобной и интуитивно понятной технологии. Рассмотрим ключевые преимущества технологии графического программирования при создании приложений на базе ЦСП.

Простой интерфейс с поддержкой Drag-n-Drop

Создание прикладных программ для ЦСП традиционно усложняется сотнями и тысячами строк кода, затрудняющими процесс отладки готовой программы. За счет использования концепции потоков данных языки графического программирования предоставляют пользователю настоящий Drag-n-Drop интерфейс, упрощают обучение и делают разработку, тестирование и отладку приложений ЦОС интуитивно понятными. На рис. 2 показан пример программы на LabVIEW. Видны две особенности:

- Первая: интерфейс пользователя и функциональная часть приложения четко разделены, это позволяет пользователю улучшить контроль разработки алгоритма. Более того, при разработке решений на основе ЦОС пользовательский интерфейс иногда играет второстепенную роль.
- Вторая: отсутствие текстового кода (готовый код на C может быть встроен в соответствующем месте).

Все функциональные блоки могут быть графическими, и потоки данных представляются в этом случае проводниками, соединяющими их. Такое преобразование приводит к очень важному преимуществу: специалисты, не являющиеся программистами, теперь могут использовать графические утилиты для программирования ЦСП. Единственное, что им необходимо знать, — это их алгоритм функционирования системы и способы его реализации на конкретной графической платформе.

Демонстрационное приложение

Для иллюстрации этого преимущества обратимся к простому примеру: разработчик хочет создать систему, которая собирает данные с аналогового входа (аудиосигнал, 44 кГц), а также применяет фильтр (фильтр Баттерворта высокой частоты третьего порядка) для удаления шума постоянного тока. Затем разработчику необходимо провести анализ входного сигнала и вывести его основные (доминантные) частоты. Данная операция довольно проста и обычно входит в некоторые ключевые концепции программирования приложений ЦОС. Кроме того, она включает в себя сбор данных с аналогового входа, фильтрацию, которая является краеугольным камнем каждого приложения ЦОС, и спектральный анализ (БПФ) в частотной области. Если разработчик захочет нарисовать блок-диаграмму для этой системы, она будет выглядеть примерно так, как показано на рис. 3.

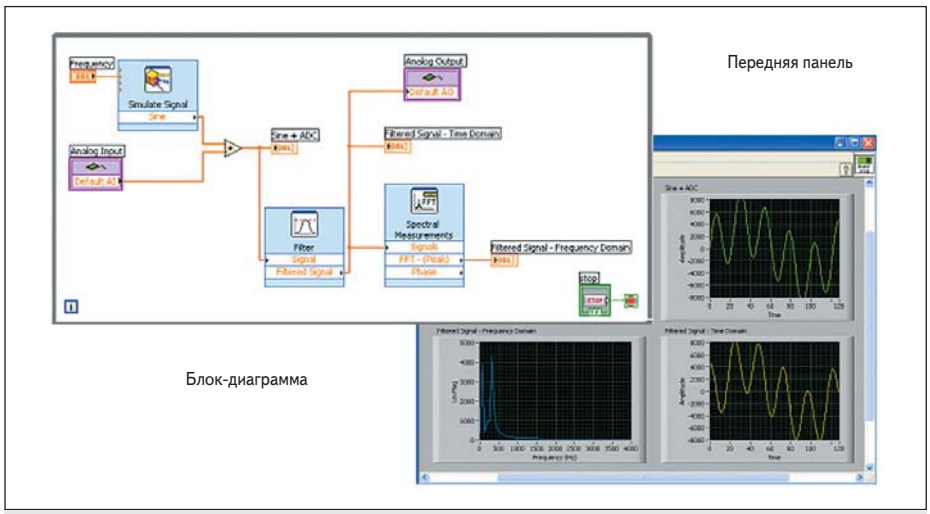

Рис. 2. Пример графической программы

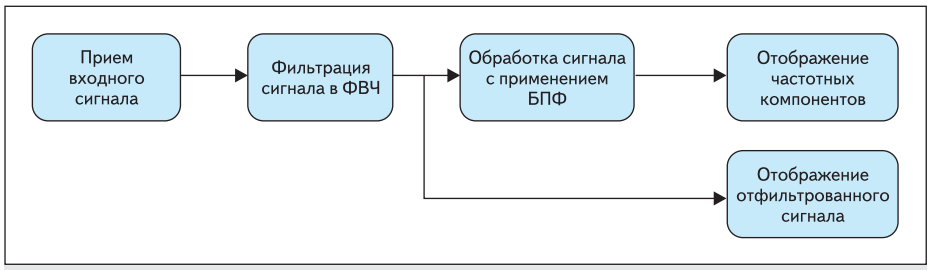

Рис. 3. Блок-диаграмма анализа входного сигнала

Используя традиционные технологии, разработчик кодирует эту блок-диаграмму с использованием C или ассемблера, а затем загружает ее в целевое устройство. Однако, используя графическое программирование, разработчик может непосредственно создать блок-диаграмму (рис. 4), которая полностью отражает алгоритм на рис. 3.

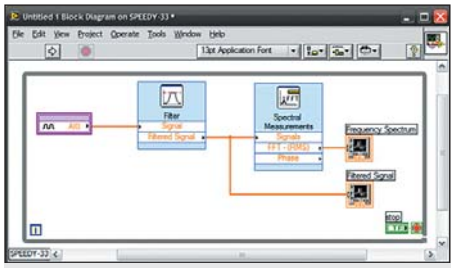

Рис. 4. Блок-диаграмма в LabVIEW для анализа входного сигнала

Теперь разработчик может не только смоделировать систему, но и протестировать ее на системном уровне, так же как при аппаратной отладке. Такой результат достигается за счет понятного блок-схемного подхода, присущего среде графического программирования.

Сотни блоков со встроенными функциями

Создание прикладных программ для ЦСП было бы неполным без использования готовых функций обработки сигналов. Большинство готовых функций можно разделить на функции обработки сигналов, фильтры, окна и функции обработки сигналов во временной и частотной областях. Как показано на рис. 2, целью графического программного обеспечения является минимизация текстового кода, необходимого для разработки алгоритма. Для этого применяются большие библиотеки встроенных функций, которые могут быть использованы для программирования ЦСП. На рис. 5 показано несколько функциональных блоков из NI LabVIEW. Похожие функции или блоки могут быть найдены и в других языках графического программирования.

Кроме блоков со встроенными функциями, языки графического программирования также предоставляют возможность использования базовых функциональных блоков для создания более сложных пользовательских

Рис. 5. Сотни блоков с готовыми функциями для ускорения разработки

функциональных блоков и их дальнейшего использования. Это очень важная возможность для групп разработчиков, создающих передовые решения на основе алгоритмов ЦОС. Специалисты, хорошо разбирающиеся в алгоритмах ЦОС, могут, используя базовые блоки, создать функциональный блок более высокого уровня, который будет исполнять более сложный алгоритм, и распространять его внутри своей команды, для того, чтобы другие разработчики могли использовать этот алгоритм в своих проектах.

Возвращаясь к демонстрационному приложению на рис. 3, можно заметить, что в целом алгоритм распадается на следующие дискретные функции:

- прием входного сигнала;
- фильтрация сигнала (высокочастотный фильтр Баттерворта 3-го порядка, БИХ);
- выполнение БПФ;
- демонстрация фильтрованного сигнала;
- демонстрация частотных компонентов.

Приложение, показанное на рис. 4, состоит из 5 блоков, каждый из которых относится к одной из указанных функций. Несмотря на то, что пример очень прост, точно так же могут быть изображены и функционально более сложные приложения ЦОС. Возможность создания блоков более высокого уровня из базовых приводит к тому, что сложные приложения ЦОС могут быть упрощены с целью повышения уровня абстракции, что, в свою очередь, ускоряет разработку и увеличивает уровень производительности системы.

Программирование с использованием конфигурируемых блоков

Ряд часто используемых функций имеет некоторые опции для их конфигурирования. В таких случаях вместо объединения 4 или 5 блоков для создания новой функции среда графического программирования позволяет использовать так называемые конфигурируемые функциональные блоки. Например, LabVIEW позволяет конфигурировать функции (Express VIs) для выполнения наиболее распространенных операций.

На рис. 6 показан пример конфигурационной панели для быстрой разработки фильтра. Этот "Express VI", как говорят продвинутые пользователи LabVIEW, дает разработчику возможность создания различных типов фильтров без написания текстового кода или просмотра множества блоков. В данном примере на рис. 6 для создания высокочастотного БИХ-фильтра с топологией Баттерворта все, что нужно разработчику, это задать необходимые параметры и частоту среза, после чего "Express VI" самостоятельно реализует фильтр, коэффициенты которого могут быть непосредственно загружены в ЦСП или экспортированы в качестве LabVIEW или C-кода.

Экспресс-блоки реализуют высокоуровневое представление, но зачастую разработчикам необходим доступ к блокам более низкого уровня. Например, для фильтра разработчик может выбрать блоки низкого уровня, с тем, чтобы создать именно тот тип фильтра, который ему необходим, с учетом специфических особенностей, например требуемой структуры.

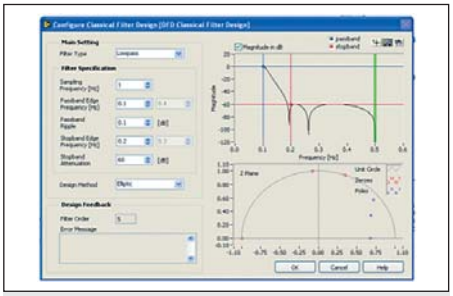

Рис. 6. Программирование на основе конфигурирования функциональных блоков для ускорения разработки

Такой подход с использованием полностью конфигурируемых блоков для создания прикладных программ для ЦСП значительно ускоряет процесс разработки, позволяя команде разработчиков всегда в срок выпускать готовую продукцию на рынок.

Возможности повторного использования кода C

Как уже было сказано в предыдущих разделах, инструменты графического программирования обеспечивают пользователя полностью графическим интерфейсом. Кроме того, некоторые разработчики специализируются на создании высокооптимизированных алгоритмов на C. Ярким примером такого алгоритма может служить быстрое преобразование Фурье (БПФ). Даже если алгоритм БПФ, реализуемый в большинстве графических окружений, эффективен, некоторые разработчики имеют специальные, оптимизированные вручную алгоритмы на C, полностью соответствующие их запросам. Среда графического программирования имеет все средства для включения в графическое приложение существующего кода на C. Например, в LabVIEW пользователь может включать существующий код C при помощи специальной структуры "C-Node". На рис. 7 показан пример генерации случайного числа и использования встраиваемой структуры "C-Node" для вывода результата на экран.

Нужно особо отметить, что компилятор, встроенный в LabVIEW, не проводит проверку на наличие ошибок в коде, который написан в "C-Node". Код просто компилируется в двоичный и затем загружается на целевое устройство. Отсюда следует, что код, встраиваемый в "C-Node", должен быть заранее проверен и отлажен, иначе вся программа будет работать неверно.

Возможность доступа к созданному C-коду

В качестве целевых платформ для среды графического программирования можно использовать любые 32-битные микропроцессорные модули или ЦСП. Для того чтобы до-

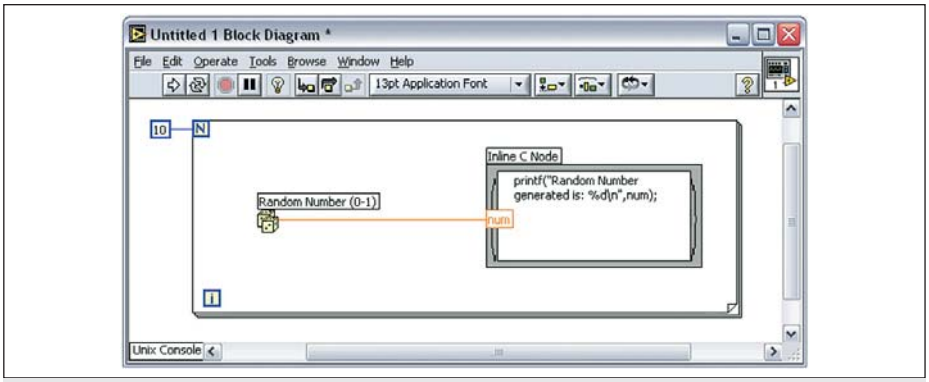

Рис. 7. Повторное использование существующих программ на языке C в графическом окружении

Рис. 8. Доступ к генерируемому С-коду

стигнуть этого, большинству утилит нужно сделать устройство распознаваемым в графической оболочке. Как только специализированное устройство подключается к оболочке, вы можете программировать его таким же образом, как делаете это со своим ПК. Такая технология работает только в том случае, если команда разработчиков использует собственные платы и соответствующие им инструменты программирования. Для поддержки этих плат и инструментов программирования большинство графических оболочек дают пользователям возможность доступа к генерируемому С-коду, который может быть использован для программирования этих плат.

На рис. 8 показано конфигурационное окно для доступа к сгенерированному С-коду в LabVIEW.

Поддержка целевых платформ различных типов

Среда графического программирования также предоставляет поддержку множества целевых платформ. Как упоминалось ранее, в среде графического программирования, в том числе и в LabVIEW, имеется возможность переноса

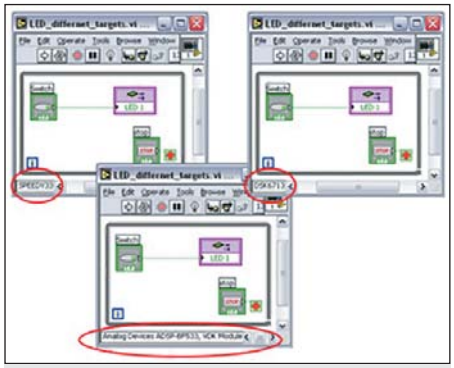

Рис. 9. Использование одинаковых приложений на различных целевых платформах

приложения с одной платформы на другую. Основным достоинством переноса является то, что одну и ту же программу можно запускать на различных платформах с минимальными изменениями кода или вовсе без них.

На рис. 9 показан пример, где одинаковые графические программы переключают светодиоды, установленные на трех разных целевых платформах ЦОС — the NI SPEEDY-33 (TI VC-33), SpectrumDigital TMS320C6713 DSK и Analog Devices ADSP-BF533 DSP.

Заключение

В последние годы сигнальные процессоры приобрели популярность среди разработчиков, благодаря их низкому энергопотреблению, низкой стоимости и высокой вычислительной мощности. В мире много профессионалов в различных областях, не специализирующихся на программировании сигнальных процессоров, но желающих использовать эту технологию в своих проектах. Специалистам в области программирования сигнальных процессоров необходимы удобные в использовании утилиты для быстрого макетирования и внедрения своих проектов. В этой статье мы рассказали о преимуществах и выгоде, которую дают технологии графического программирования, предоставляя надежную платформу для разработки, макетирования и внедрения систем с применением ЦСП. Системы графического программирования, в том числе LabVIEW, являются основными инструментами, позволяющими снизить время на разработку проектов и быстрее выйти на рынок за счет использования встроенных готовых функций и программирования на основе конфигурируемых экспресс-блоков. Системы графического программирования поддерживают широкий спектр устройств, являясь, таким образом, идеальной платформой для макетирования, позволяющей определить оптимальное целевое устройство для конкретного приложения.

Литература

1. Benveniste A., Le Guernic P. Hybrid Dynamical Systems Theory and the SIGNAL Language. IEEE Tr. on Automatic Control, Vol. 35, No. 5, May 1990.
2. Halbwachs N., Caspi P., Raymond P., Pilaud D. The Synchronous Data Flow Programming Language LUSTRE. Proceedings of the IEEE, Vol. 79, No. 9, 1991.
3. Hilfinger P. A High-Level Language and Silicon Compiler for Digital Signal Processing. Proceedings of the Custom Integrated Circuits Conference, IEEE Computer Society Press. Los Alamitos, CA 1985.
4. The Ptolemy Project http://ptolemy.eecs.berkeley.edu/
5. Overview of LabVIEW http://zone.ni.com/devzone/conceptd.nsf/webmain/F34045D2CC5357F486256D3400648C0F?OpenDocument&node=200067_us