Проектирование с использованием процессоров Analog Devices. Введение в VisualDSP++

Александр СОТНИКОВ,
к. т. н.
alexander.sotnikov@analog.com.ru

В этой статье мы начнем знакомиться со средой разработки и отладки программного обеспечения VisualDSP++. Она носит вводный характер и посвящена вопросам установки и лицензирования VisualDSP++, а также общим принципам работы над проектами.

Установка и первый запуск VisualDSP++

Как уже отмечалось в предыдущей статье [1], интегрированная среда VisualDSP++ является основным средством разработки и отладки приложений для процессоров компании Analog Devices. Последняя версия VisualDSP++ имеет номер 5.0, работает под ОС Windows 2000 SP4, Windows XP SP2, Windows Vista или Windows 7 Business/Enterprise/Ultimate edition и поддерживает семейства процессоров TigerSHARC, SHARC и Blackfin. Она может быть загружена с веб-сайта компании (http://www.analog.com/processors/testdrive) после бесплатной регистрации. По завершении регистрации пользователю высылается серийный номер, дающий возможность работать с пакетом VisualDSP++ в тестовом режиме в течение 90 дней. Также VisualDSP++ с тестовой лицензией можно получить на компакт-диске у региональных дистрибьюторов или заказать через веб-сайт компании Analog Devices.

При запуске установочного файла, загруженного с веб-сайта или находящегося на компакт-диске, вызывается программа установки, которая распаковывает все необходимые для работы VisualDSP++ файлы и драйверы, а также примеры программ и файлы справочной системы, в указанный пользователем каталог (по умолчанию это каталог Program Files\Analog Devices\VisualDSP 5.0\ на системном диске). Кроме того, программа установки автоматически создает папку **Программы → Analog Devices → VisualDSP++ 5.0** в меню **Пуск ОС Windows**.

Новые версии VisualDSP++ выпускаются достаточно редко (раз в 3–4 года), поэтому для исправления обнаруженных ошибок в программе, а также добавления в нее поддержки новых кристаллов и оценочных плат процессоров разработчики пакета регулярно выпускают обновления (апдейты), которые также можно загрузить через Интернет с веб-сайта Analog Devices. Загруженный файл обновления устанавливается поверх текущей версии VisualDSP++ при помощи пункта **Программы → Analog Devices → VisualDSP++ 5.0 → Maintain this installation** меню **Пуск ОС Windows**. Файлы обновлений кумулятивны, то есть, например, файл обновления с порядковым номером 7 включает в себя все исправления и дополнения, которые присутствуют в файлах с порядковыми номерами с 1-го по 6-й.

При первом запуске программа VisualDSP++ выдает предупреждение о том, что не найдена необходимая лицензия. Нажатие кнопки «Да» вызывает диалоговое окно управления лицензиями (рис. 1). Лицензия VisualDSP++ бывает двух типов: фиксированная (node-locked) и плавающая (floating). Фиксированная лицензия, которая привязана к конкретному пользователю, в свою очередь, бывает как платной (полная лицензия), так и бесплатной (тестовая лицензия и лицензия, поставляемая с платой EZ-KIT/EZ-BRD). Для установки этого типа лицензии в диалоговом окне необходимо ввести серийный номер, полученный при регистрации на веб-сайте Analog Devices или указанный на компакт-диске с VisualDSP++. Плавающая лицензия бывает только платной. Она предназначена для работы в сети и состоит из двух частей: серверной (для нее, как и в случае с фиксированной лицензией, вводится серийный номер) и клиентской (для которой необходимо указать путь к файлу лицензии на сервере). Для нормальной работы с платными лицензиями и лицензией, поставляемой с EZ-KIT/EZ-BRD, серийный номер необходимо зарегистрировать. В противном случае по истечении определенного периода времени программа перестанет запускаться. После регистрации серийного номера пользователю высылается код подтверждения (validation code), ввод которого снимает временные ограничения. Вызов процедуры регистрации и ввод серийного номера также выполняется через диалоговое окно управления лицензиями, которое может быть открыто в любой момент через пункт **Программы → Analog Devices → VisualDSP++ 5.0 → Manage Licenses** меню **Пуск ОС Windows**, через меню VisualDSP++ или при запуске Visual DSP++ с нажатой клавишей Ctrl. В заключение обзора вопросов лицензирования следует отметить, что, несмотря на поддержку средой сразу трех семейств процессоров, для каждого из них требуется установка отдельной лицензии.

Рис. 1. Диалоговое окно управления лицензиями

После настройки корректной лицензии открывается окно графического интерфейса VisualDSP++. Оно, как показано на рис. 2, включает в себя:

- строку заголовка;
- строку меню, панель инструментов и строку состояния;
- окно проектов (Project Window);
- окно вывода информации (Debug Window).

Назначение строк заголовка, меню, панели инструментов и строки состояния очевидно, и на данном этапе обсуждать его не будем.

Окно проектов предназначено для работы с проектами, внутри которых ведется вся разработка программного обеспечения в VisualDSP++. С его помощью пользователь добавляет исходные файлы в проект, переключается между проектами, настраивает параметры проекта и утилит проектирования.

Окно вывода информации используется для отображения различных текстовых информа-

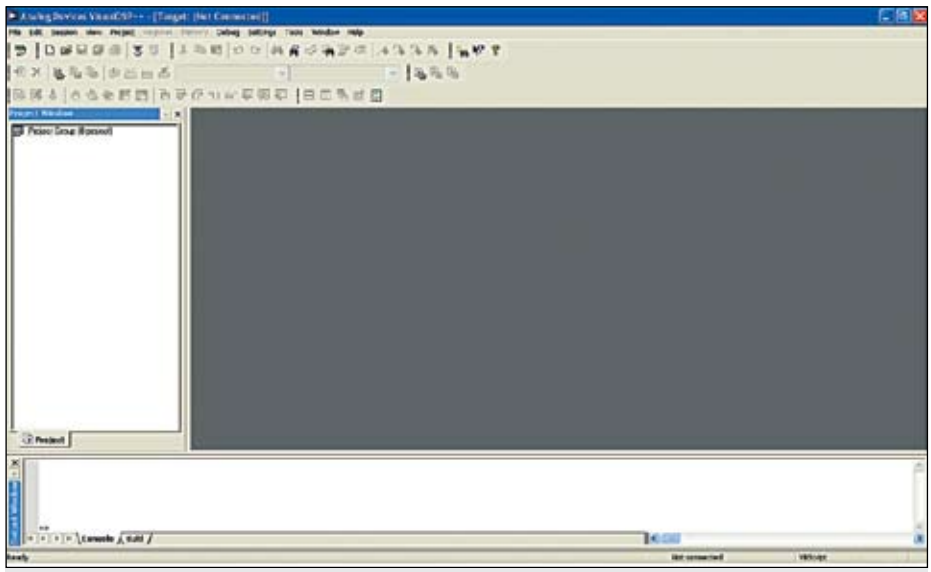

Рис. 2. Окно графического интерфейса VisualDSP++

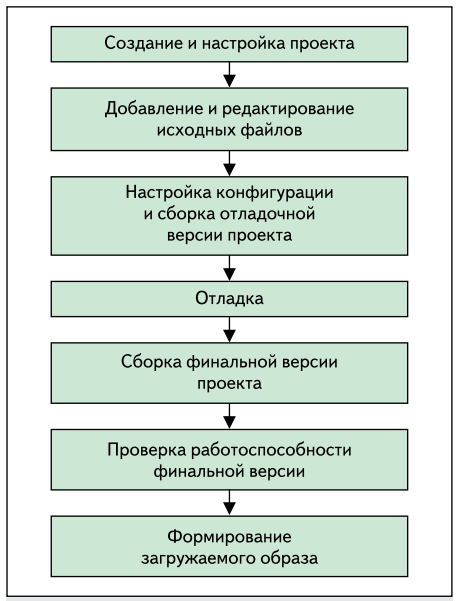

Рис. 3. Этапы работы над проектом

ционных сообщений и сообщений об ошибках VisualDSP++ и утилит разработки, а также представляет собой интерфейс для ввода пользовательских команд и загрузки скриптов.

В главном окне VisualDSP++ также могут быть открыты окна редактора файлов и различные отладочные окна, в которых выводится содержимое памяти, регистров, значений переменных и другая информация, упрощающая отладку приложений. Кроме того, в главном окне VisualDSP++ выводятся окна вспомогательных графических утилит, таких как Expert Linker или FlashProgrammer.

Мы еще рассмотрим подробно каждый из элементов графического интерфейса VisualDSP++ в последующих статьях, а сейчас, прежде чем перейти к созданию первой программы, посвятим некоторое время вопросам терминологии и обсуждению общих принципов разработки приложений в среде VisualDSP++.

Терминология VisualDSP++ и основные этапы работы над проектом

Основной структурной единицей в VisualDSP++ является проект — совокупность исходных файлов программы и настроек утилит среды, которые используются при создании программы, предназначенной для исполнения процессором (или многопроцессорной системой). Вся информация о проекте хранится в VisualDSP++ в файле с расширением .dpj. Проект, в большинстве случаев, разрабатывается под конкретную аппаратную платформу. Доступ среды к аппаратной платформе осуществляется через отладочные средства (симулятор, эмулятор или технология Debug Agent (отладочный агент)), которые в терминологии VisualDSP++ называются целевыми объектами (Target) или целевыми объектами отладки (Debug Target). В зависимости от выбранного типа целевого объекта платформой могут являться:

- один или несколько однотипных процессоров при работе с симулятором;
- процессор на плате EZ-KIT Lite/EZ-BRD при работе через отладочный агент;
- один или несколько процессоров (в общем случае разного типа), объединенных в цепочку JTAG при работе с эмулятором.

Подключение к платформе для загрузки и отладки исполняемого файла осуществляется в сессиях. В отдельно взятый момент времени может быть открыта только одна сессия, и, например, для перехода от моделирования к внутрисхемной отладке или отладке в составе оценочной платы необходимо выйти из одной сессии и открыть новую. Для настройки любых сессий, за исключением тех, в которых используется симулятор, в состав VisualDSP++ включена утилита VisualDSP++ Configurator. Доступ к ней осуществляется через пункт **Программы → Analog Devices → VisualDSP++ 5.0 → VisualDSP++ Configurator** меню **Пуск ОС Windows**.

Наиболее типичный процесс работы над проектом в среде VisualDSP++ включает в себя следующие этапы (рис. 3).

На первом этапе пользователь создает файл проекта (.dpj) и настраивает его параметры (тип процессора, настройки ассемблера, компилятора, компоновщика и т. д.). Для упрощения создания проектов в состав VisualDSP++ входит утилита под названием Project Wizard, которая позволяет в интерактивном режиме настроить параметры проекта, а также автоматически сгенерировать ряд вспомогательных файлов, включая файл описания линкера и run-time заголовок C, с учетом специфики используемой платформы.

При желании на этапе создания проекта пользователь может включить в создаваемый проект поддержку ядра операционной системы реального времени VDK (VisualDSP++ Kernel). VDK — это масштабируемая программная исполняемая система, жестко интегрированная с VisualDSP++, использование которой позволяет абстрагироваться при создании ПО от деталей аппаратной реализации и сосредоточиться на написании алгоритмов обработки данных. VDK предоставляет пользователю средства планирования и управления ресурсами, специфичные для адресации памяти и временных ограничений при программировании ЦСП Analog Devices. Применение этих средств может повысить эффективность разработки сложных проектов, требующих параллельного исполнения большого числа процессов.

При работе с процессорами Blackfin пользователи могут создавать проекты с поддержкой сетевого стека с открытым исходным кодом LwIP. Эта возможность также выбирается пользователем на этапе создания проекта.

Следующий этап заключается в добавлении в проект существующих или создании новых исходных файлов. Исходные файлы программы могут быть написаны на языке ассемблера процессора (имеющие расширение .asm, .s или .dsp) и/или высокоуровневых языках программирования C/C++ (.c/.cpp/.cxx). Оба этих варианта имеют свои достоинства и недостатки. Преимущества высокоуровневых языков заключаются в модульности, переносимости на другие платформы и простоте повторного использования кода. Манипуляции данными в языках C/C++ обычно выполняются на высоком уровне абстракции путем использования переменных и вызовов функций/процедур, что делает программу более понятной. В свою очередь операции с данными при программировании на языке ассемблера осуществляются на уровне реальных регистров, вычислительных блоков и блоков памяти процессора. Написание программ на языке ассемблера — это сложный, занимающий много

времени процесс, который требует от программиста хороших навыков программирования и повышенной внимательности. Однако несомненным достоинством ассемблера является повышенные компактность и быстродействие кода, поскольку ни один, даже самый продвинутый компилятор не способен генерировать код, превосходящий по этим показателям грамотно написанную вручную программу на языке ассемблера. Именно поэтому разработчики зачастую используют в своих проектах комбинацию языков C/C++ и ассемблера. Высокоуровневые языки отлично подходят для задач управления и базовых манипуляций данными, а ассемблер идеален для интенсивных числовых расчетов и выполнения задач, критичных к времени обслуживания.

Также на данном этапе в проект могут быть добавлены (если это не было сделано автоматически при помощи Project Wizard) два вспомогательных файла: файл run-time заголовка и файл описания линкера. Файл run-time заголовка C (CRT header) необходим в случае, если основная часть программы написана на языке C/C++. Он представляет собой код на языке ассемблера, который выполняется после перехода процессора по начальному адресу программы при включении питания или сбросе. Этот код приводит процессор в известное состояние и вызывает функцию main. Выполнение run-time заголовка гарантирует, что при входе в main-состояние процессор подчиняется двоичному интерфейсу приложений C (ABI), а глобальные данные, объявленные в приложении, инициализированы соответственно требованиям стандартов C/C++. Файл описания линкера (Linker Description File, LDF) используется для управления процессом компоновки исполняемого кода. Если один или оба этих файла к проекту в явном виде не добавляются, то в процессе компоновки используются стандартные файлы для выбранного процессора. Также допускается добавление в проект и иных типов файлов, однако при сборке проекта будут обработаны только те файлы, которые распознаются утилитами VisualDSP++.

Когда все исходные и вспомогательные файлы готовы, можно приступить к сборке проекта. Этот процесс происходит в два шага. Сначала исходные файлы и CRT подвергаются компиляции и ассемблированию при помощи утилит «компилятор» и «ассемблер» соответственно. В результате формируется один или несколько объектных файлов в стандартном формате ELF (Executable and Linkable Format), которые имеют расширение .doj и содержат в себе структурированные секции кода и данных. Также в процессе ассемблирования может быть сгенерирован необязательный файл листинга (.lst), содержащий справочную информацию о результатах ассемблирования. Затем вызывается утилита «линкер» (компоновщик), которая на основании информации, содержащейся в файле описания линкера (.ldf), анализирует объектные файлы и подключаемые пользователем библиотеки (.dlb) и определяет, какие из объектов необходимо поместить в тот или иной сегмент доступной процессору внутренней или внешней памяти. Результатом работы линкера является исполняемый файл в формате ELF с расширением .dxe. В некоторых случаях линкер также формирует файлы содержимого совместно используемой памяти (для проектов, исполняемых в многопроцессорных системах или на двуядерном процессоре Blackfin ADSP-BF561) с расширением .sm и файлы оверлейной памяти (для систем, в которых отдельные независимые части кода оперативно подгружаются из внешней памяти во внутреннюю память процессора) с расширением .ovl.

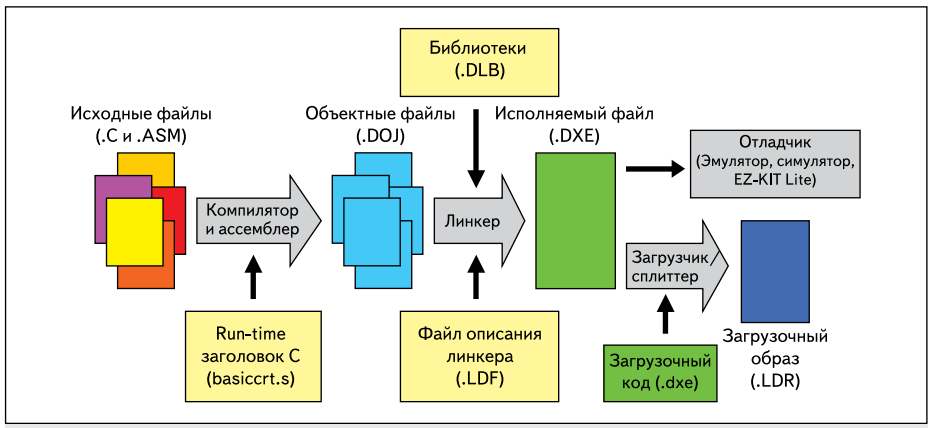

Рис. 4. Основные типы файлов и утилиты VisualDSP++, используемые при работе над проектом

Все необходимые для формирования исполняемого файла утилиты запускаются средой VisualDSP++ автоматически в фоновом режиме после выбора пользователем соответствующего пункта меню или нажатия кнопки на панели инструментов, а информация о ходе процесса сборки проекта, предупреждения и сообщения об ошибках выводятся в окне Debug Window. Как правило, на данном этапе сборка проекта происходит в конфигурации Debug, в которой отключена оптимизация компилятора и в код добавляются возможности отладки.

В случае успешной сборки проекта пользователь настраивает отладочную сессию, после чего исполняемый файл загружается через целевой объект в программную модель процессора или в реальный процессор для отладки.

Когда проект успешно проходит функциональную отладку, он может быть повторно собран в конфигурации Release. Эта конфигурация дает оптимизированный по производительности код без возможностей отладки (или с ограниченными возможностями). Работоспособность полученного в результате сборки исполняемого файла необходимо повторно исследовать в составе платформы.

Поскольку формат исполняемого файла не годится для непосредственной загрузки в процессор в автономно работающей системе, он должен быть преобразован утилитой loader (загрузчик) в загружаемый образ (.ldr). В ряде случаев (например, когда часть кода и/или данных должна быть помещена в процессе загрузки во внешнюю статическую или динамическую память) перед загрузкой процессора необходимо проинициализировать его аппаратные средства, такие как контроллер внешней памяти или регистры формирования сигнала внутренней тактовой синхронизации. Для этих целей на этапе создания загружаемого образа к основному исполняемому файлу добавляется вспомогательный исполняемый файл, называемый инициализационным кодом. Если процедура загрузки процессора в системе не требуется, а программа будет исполняться непосредственно из внешней ПЗУ, то для формирования файла с кодами команд используется утилита сплиттер. (Для процессоров семейства Blackfin функции загрузчика и сплиттера выполняет одна утилита.) Полученный в результате работы загрузчика/сплиттера файл может быть записан в микросхему памяти на плате EZ-KIT/EZ-BRD или плате собственной разработки при помощи утилиты Flash Programmer, которая предназначена для внутрисхемного программирования микросхем памяти непосредственно из среды VisualDSP++.

Основные используемые в процессе работы над проектом типы файлов и утилиты VisualDSP++ показаны на рис. 4.

Заключение

Среда VisualDSP++ обладает всеми средствами, необходимыми для максимально быстрого создания приложений на базе процессоров компании Analog Devices. В этой статье дан лишь краткий ее обзор. Более подробно аспекты проектирования в среде VisualDSP++ будут рассматриваться на практических примерах в последующих статьях цикла.

Литература

1. Сотников А. Средства разработки и отладки программного обеспечения для процессоров Analog Devices // Компоненты и технологии. 2010. № 2.