ЗАО АВТЭКС Санкт-Петербург тел/факс: 567-72-02 info@autex.spb.ru

РЕАЛИЗАЦИЯ FRAMEWORK ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ, ПОСТРОЕННЫХ НА АППАРАТНОЙ ПЛАТФОРМЕ DSP TI 64XX

Машин А.П.[1], Цивинский М.Ю.[1], Behan H.[2]

[1]«ООО ИнтерТел Сибирь», Россия; [2]STROM Telecom s.r.o., Czech Republic

Разрабатываемое нашими компаниями, программное обеспечение применяется для обработки сигналов в телекоммуникационных системах. Типовые задачи цифровой обработки сигналов (ЦОС) в этой области это кодеки IP телефонии, эхоподавление, распознавание речи, обработка частотных сигнализаций, модемные алгоритмы и т.п. Дополнительно цифровые сигнальные процессоры (DSP) могут выполнять и нехарактерные для них функции коммуникационных процессоров, такие как обработка протоколов SS-7 и X.25. В общем, программное обеспечение DSP должно выполнять следующие задачи:

- параллельная обработка нескольких десятков и даже сотен каналов, разговорных трактов, на одном DSP;
- параллельное функционирование различных алгоритмов ЦОС, работающих совместно на одном DSP процессоре, и поддержка гибкого взаимодействия между ними;
- интенсивный обмен сообщениями, данными и командами между DSP процессорами на многопроцессорной плате, или между DSP и ведущим (HOST) процессором;
- обеспечение функционирования системных процессов, как-то драйверы интегрированной и внешней периферии, стеки коммуникационных протоколов и т.п.

Исходя из приведенных задач, очевидно, что используемые сигнальные процессоры должны обладать высокой производительностью, в своих разработках мы используем DSP процессоры семейства Texas Instruments C64xx, а программное обеспечение должно использовать многопоточную операционную систему реального времени (ОС) в сочетании с Framework, обеспечивающим решение столь разнородных задач. Используемый в разрабатываемых нами проектах Framework, является разработкой специалистов нашей компании и ориентирован на решение перечисленных выше задач.

Структурная схема Framework приведена на Рисунке 1. Здесь представлен только верхний уровень основных базовых компонент – не показаны системные элементы Framework, такие как модуль инициализации, общий каркас и некоторые другие. На этом рисунке также не показаны различные потоки (threads), в рамках которых выполняется работа того или иного модуля. Это связано с тем, что в одном модуле могут параллельно существовать несколько потоков или наоборот, один поток охватывать несколько модулей, все это зависит от конкретного назначения модуля и может варьироваться в зависимости от требований задачи. Структурно Framework состоит из следующих компонент:

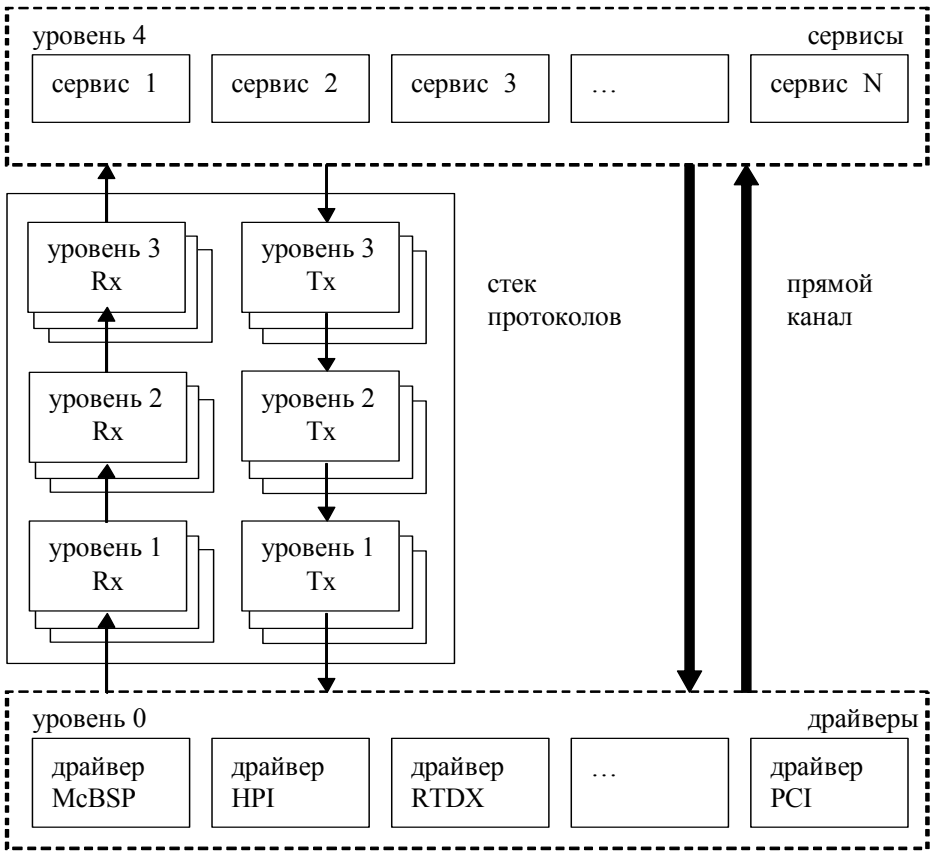

Рисунок 1

1. Сервисы – модули, отвечающие за выполнение определенного алгоритма ЦОС, или группы алгоритмов, если это логически обусловлено требованиями задачи. Каждый сервис, как правило, отвечает за обработку нескольких каналов, в реальных проектах число каналов может доходить до нескольких сотен. В типичном случае, сервис принимает поток цифровых данных из драйвера периферийного устройства, производит обработку по нескольким каналам и передает результат обработки в исходящий поток. Кроме вычислительной работы сервис может производить обмен сообщениями и блоками данных с другими сервисами, подсистемами Framework, с другим DSP или HOST процессором. Сервис может быть подключен в структуру Framework или переконфигурирован динамически. Для разработчиков приложений Framework предоставляет базовые конструкции для создания конкретного сервиса, при этом разработчик сервиса может иметь лишь минимальное представление о структуре и реализации Framework. Подключение сервиса и регистрация его в структуре Framework производится автоматически, что позволяет эффективно разделить разработку между прикладными программистами, уровень Сервиса, и системными, собственно сам Framework.

2. Многоуровневый стек протоколов, для обмена данными между DSP процессорами, или DSP процессорами и HOST процессором, использующий несколько типов сред передачи данных. Стек коммуникационных протоколов в Framework представляет собой открытую систему, которая позволяет системному программисту разрабатывать различные, в зависимости от задачи, уровни протоколов, используя уже созданный базовый каркас. Структура стека протоколов представляет собой трехмерную конструкцию, каждый уровень которой может быть представлен несколькими специфичными, параллельно работающими реализациями. Уровень 1, согласно рисунку 1, транспортный уровень, отвечает за прием пакета данных от конкретного драйвера физического канала связи. При необходимости работы с различными средами, по сути, с различными драйверами, создается несколько объектов транспортного уровня. На сегодняшний день разработаны транспортные уровни для последовательной связи (McBSP), разделяемой памяти (HPI), внутрисхемного эмулятора (RTDX), интегрированной PCI шины, HDLC и модемных протоколов. Приняв блок данных от драйвера, транспортный уровень обрабатывает заголовок блока, трейлер и выделяет из принятого блока вложенный пакет данных, который передается Уровню 2, пакетному уровню. Пакетный уровень отвечает за достоверность принятых данных, проводит коррекцию ошибок, принимает решение о дальнейшей обработке пакета данных и передает инкапсулированные в пакете данные на Уровень 3, сервисный уровень. В одном пакете может содержаться несколько сообщений сервисного уровня, адресованных, возможно, различным сервисам, поэтому сервисный уровень проводит разделение сообщений, сортировку по приоритетам и передает их непосредственно сервисам. В системе

обычно присутствует несколько специализированных объектов сервисного уровня, предназначенных для обмена короткими сообщениями, большими блоками данных или с установлением и поддержкой соединения (сессии). Предусмотрена возможность связи между параллельными уровнями, например между Уровнем 2 на прием и на передачу, что может потребоваться для тестовых приложений.

3. Драйверы предназначены для взаимодействия с периферийными устройствами DSP. Framework позволяет использовать совместно драйверы собственной реализации и драйверы разработанные согласно концепции GIO\IOM Drivers, которую предлагает Texas Instruments. При этом GIO-интерфейс легко совмещается с используемой в Framework системой передачи сообщений. Кроме GIO-интерфейса возможно сделать сопряжение и с двумя другими моделями: PIP и SIO, такая совместимость позволяет использовать стандартные драйверы от сторонних разработчиков в собственном Framework. Драйверы могут взаимодействовать с сервисами через стек коммуникационных протоколов либо напрямую, рисунок 1 – «прямой канал». Прямой канал драйвера с сервисом используется в основном для пересылки цифровых потоков реального времени, например, голосовых каналов, в этом случае обеспечивается максимальная эффективность процесса передачи данных. Распределение потоков данных между сервисами может быть реализовано либо собственно в драйвере, либо на дополнительном подуровне, надстройке над драйвером, мультиплексоре/демультиплексоре. Введение этого подуровня обусловлено необходимостью обработки одного потока данных нескольким различными алгоритмами ЦОС параллельно и независимо друг от друга.

4. Многозадачная операционная система реального времени. В качестве операционной системы была выбрана DSP/BIOS, поставляемая в комплекте с Code Composer Studio IDE, разработки Texas Instruments. DSP/BIOS имеет практически все необходимые базовые механизмы для создания полноценной ОС с учетом специфики DSP. Реально, DSP/BIOS можно рассматривать как конструктор, из которого возможно выбрать необходимые элементы и настроить их для конкретной задачи. Выбор DSP/BIOS обусловлен встроенной в нее поддержкой части интегрированной в DSP периферии. Предполагается, что в дальнейшем вместо прямого использования API DSP/BIOS все вызовы функций ОС будут использовать POSIX-подобный интерфейс, что позволит использовать совместно с Framework любую другую подходящую по характеристикам ОС реального времени.

5. Общий каркас, Framework, который объединяет перечисленные выше компоненты и механизм передачи сообщений. Framework полностью реализован с использованием C++, при этом ряд базовых компонентов создан с использованием методологии Design Patterns и применением перенесенного нами на DSP платформу C++ STL. Интерфейс между любыми объектами, входящими в состав Framework, осуществляется с помощью унифицированного механизма передачи сообщений, представляющей собой развитие таких шаблонов проектирования (Design Patterns) как «Typed Message» и «Multicast». Обмен данными между уровнями в стеке протоколов также производится с помощью унифицированного механизма обмена сообщениями. Используемый механизм имеет следующие особенности:

- возможность передачи отправителем сообщений любых типов; приемник может иметь несколько обработчиков для различных типов сообщений;
- возможность использования как асинхронного, так и синхронного способа передачи сообщений;
- большая гибкость и универсальность, в частности, данный механизм может быть легко адаптирован для совместного использования с подобными механизмами от сторонних разработчиков.

Из опыта проведенной нашими специалистами работы можно сделать следующие выводы:

- Создание проектов на современном уровне требует применения ОС реального времени. Использования ОС реального времени для DSP позволяет существенно повысить сложность разрабатываемых систем и, соответственно, расширить класс решаемых на них задач. По нашему убеждению, использование полноценной ОС для реализации систем на DSP класса TI C64xx является обязательным условием, а не общей рекомендацией (разумеется, возможны специфичные задачи).
- Увеличение потребляемых ресурсов, при переходе к использованию ОС, языков программирования высокого уровня (C++, C++ STL, templates), не является критичным для современных DSP процессоров. Тем самым реализуется эффективное повторное использование кода и достигается высокая скорость разработки. Дополнительно появляется возможность привлекать к разработке программного обеспечения DSP специалистов высокого уровня по программированию микропроцессоров общего назначения.
- Нередко, управляющие микропроцессоры встраиваемых систем имеют сравнимую с DSP производительность, часто, существенно меньшую, по сравнению с совокупной производительностью всех DSP установленных в оборудовании. Достигнутая современными DSP производительность, создание эффективных компиляторов языков высокого уровня и богатая интегрированная периферия позволяют DSP эффективно решать задачи ранее им несвойственные. Многие задачи могут быть решены вообще без применения дополнительной периферии и управляющих процессоров общего назначения, например, обработка стека TCP/IP с взаимодействием по сети Ethernet, взаимодействие с другими устройствами через шину PCI, вывод графической информации и т.п.
- Четкая модульная конструкция Framework, благодаря возможности разделения работ, позволяет значительно сократить время разработки и тестирования конечного проекта, а также позволяет осуществлять перенос ее на другие аппаратные платформы, предназначенные для решения подобных задач, например на коммуникационные процессоры, или процессоры общего назначения.

Литература

1. Применение шаблонов проектирования. Дополнительные штрихи. Джон Влиссидес. М,. Издательский дом «Вильямс», 2003.
2. Язык программирования C++. Специальное издание. Бьерн Страуструп. М., «Издательство БИНОМ», 2001.
3. SPRU 616. DSP/BIOS Driver Developer’s Guide. Texas Instruments. 2002.
4. Современное проектирование на C++. Обобщенное проектирование и прикладные шаблоны проектирования. Андрей Александреску. М., «Издательский дом Вильямс», 2002.