

# Создание сложных VDK /LwIP приложений на процессорах Blackfin

Компания Analog Devices портировала облегченный стек TCP/IP (LwIP, light-weight TCP/IP) на семейство встраиваемых процессоров Blackfin. Стек LwIP может использоваться в процессорах Blackfin для разработки аудио/видеоприложений или приложений промышленной автоматизации с поддержкой сетевого взаимодействия. LwIP — это интересный программный пакет, который позволяет быстро преобразовывать автономное встраиваемое приложение в сетевое.

Сангай КУШАЛ

Для создания высокоэффективного и устойчивого приложения в процессе интеграции необходимо предпринять ряд мер. В этой статье даются рекомендации по интеграции LwIP с другими периферийными узлами или программными компонентами системы, следование которым позволит избежать большинства типичных ошибок при разработке приложений на базе LwIP на процессорах Blackfin. Также для облегчения построения эффективной системы в статье обсуждаются используемые при реализации стека LwIP ресурсы процессора.

## Мотивация

В последнее время встраиваемые сетевые приложения все чаще становятся ориентированными на передачу мультимедийного содержимого по проводным или беспроводным сетям. Поддержку сетевого взаимодействия также желательно иметь в приложениях, относящихся к таким областям, как промышленная автоматизация и управление. На рис. 1 показаны типовые приложения, в которых желательна возможность подключения к интернету.

Компанией Analog Devices разработаны бесплатные (распространяемые на условиях лицензионных соглашений) программные составные блоки, помогающие ускорить создание сложных встраиваемых сетевых приложений. Далее приводится список базовых программных компонентов, поставляемых в составе среды разработки программного обеспечения процессоров Analog Devices VisualDSP++ версий 4.5 и выше:

- LwIP — облегченный стек TCP/IP [1], портированный на процессоры ADSP BF53x и ADSP BF56x<sup>1</sup>.

- Драйверы устройств и библиотеки системных служб (SSL, system services libraries) — API-функции для конфигурирования системных ресурсов и периферийных устройств и управления ими [10].
- VisualDSP++ Kernel (VDK) [13] — ядро реального времени. Это нетребовательное к ресурсам ядро может использоваться для эффективного управления системными ресурсами в среде многозадачного приложения.
- MM SDK-комплект для разработки мультимедийного программного обеспечения (Multimedia Software Development Kit) [14], который включает в себя примеры программ для реальных мультимедийных приложений.

Разработка реального встраиваемого сетевого приложения может потребовать интеграции этих программных составных блоков в единую систему. При интеграции разнородных программных компонентов возможны конфликты программных и аппаратных ре-

сурсов. Кроме того, использование программных компонентов со столь высоким уровнем абстракции на имеющей ограниченные ресурсы встраиваемой платформе может приводить к неоптимальному использованию системных ресурсов или к некорректному управлению ими.

## Обзор LwIP

Стек LwIP представляет собой облегченную реализацию стека протоколов TCP/IP. Это бесплатное программное обеспечение с открытым исходным кодом, которое можно загрузить по ссылке <http://savannah.nongnu.org/cgi-bin/viewcvs/LwIP/LwIP/>. Стек LwIP не требует большого объема памяти и поэтому хорошо подходит для встраиваемых приложений. Подробную информацию об LwIP можно найти в Интернете по ссылке <http://www.sics.se/~adam/lwip/>.

Стек LwIP портирован на процессоры ADSP BF53x и ADSP BF56x и входит в состав

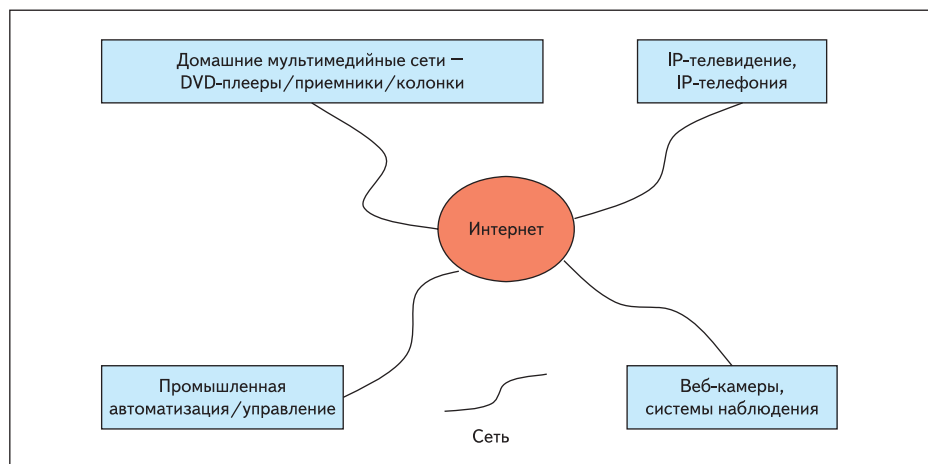


Рис. 1. Типичные встраиваемые сетевые приложения

<sup>1</sup> На процессоры Blackfin ADSP BF535 стек LwIP не портирован.

среды VisualDSP++. Можно начать разрабатывать TCP/IP приложения на процессорах Blackfin с помощью оценочной системы ADSP BF537 EZ KIT Lite, которая имеет встроенный интерфейс Ethernet. Для процессоров ADSP BF533/BF532/BF531 и ADSP BF561 может использоваться плата расширения USB LAN. Информацию, необходимую для того, чтобы начать разрабатывать проекты с поддержкой TCP/IP, можно найти в LwIP User Guide (руководство пользователя по LwIP) [1]. В реализации LwIP на процессорах Blackfin используется ядро VisualDSP (VDK, VisualDSP kernel) и библиотеки системных служб (SSL) и драйверов устройств (DD). Информацию о VDK и библиотеках системных служб можно почерпнуть из публикаций [10] и [7].

### Требования к ресурсам для работы LwIP на процессорах Blackfin

#### Пропускная способность LwIP на процессорах Blackfin

Пропускная способность портированного на процессоры Blackfin стека LwIP измерялась при помощи утилиты ADI TTCP [21], построенной на основе бесплатной утилиты PCATTCP.

На рис. 2 показана зависимость пиковой пропускной способности от размера буфера для реализации LwIP на платформе ADSP BF537. Реальные результаты могут отличаться от приведенных на рис. 2 и зависеть от сетевого трафика и скорости, поддерживаемой сетью. Как показано на рис. 2, поддерживаемая LwIP пропускная способность для стеков протоколов TCP и UDP достигает 11 Мбайт/с.

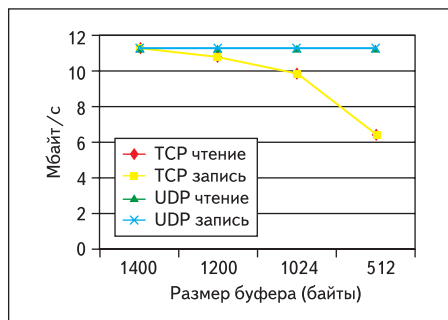


Рис. 2. Зависимость пиковой пропускной способности от размера буфера

#### Требования к памяти для LwIP на процессорах Blackfin

Как уже отмечалось ранее, в портированном на Blackfin стеке LwIP используется VDK и библиотеки системных служб (SSL). В таблице 1 указаны объемы памяти для каждого из программных компонентов LwIP-приложения в отдельности, что позволяет получить лучшие оценки при интеграции проектов, в которых уже используется один из этих компонентов. Приведенные цифры являются приблизительными и соответствуют про-

Таблица 1. Распределение кода и данных в памяти для приложения на базе LwIP

Программный компонент (протокол)	UDP (код), кбайт	TCP (код), кбайт	UDP (данные), кбайт	TCP (данные), кбайт
LwIP	35	52	10	12
Библиотеки системных служб	15	15	1,2	1,2
Библиотеки VDK + другие компоненты	28	28	43	43
Всего	78	95	54,2	56,2

Таблица 2. Требования к объему «кучи» для LwIP

Тип приложения	Объем буферов драйверов, кбайт	Объем «кучи», кбайт	Пул памяти, кбайт	Буферы пакетов, кбайт	Полный объем, кбайт
Стандартное приложение, создаваемое по умолчанию	Rx = 8×1560 Tx = 2×1548 ~16	~64	~7	~64	~151
Большие приложения с пакетами большого размера (аудио/видеоприложения)	Rx = 16×1560 Tx = 8×1548 ~50	~128	~7	~128	~313
Небольшие приложения с пакетами малого размера (задачи управления)	Rx = 8×256 Tx = 8×256 ~4	~8	~7	~8	~27

стой программе Hello world, исполняемой в стеке LwIP.

#### Другие необходимые системные ресурсы Прерывания и DMA

При исполнении стека LwIP на процессорах ADSP BF537 драйверы работают со встроенным контроллером Ethernet. По умолчанию прерывания каналов DMA Ethernet отображены в системное прерывание IVG11, которое не используется другими каналами DMA.

На процессорах ADSP BF533/BF532/BF531 и ADSP BF561 стек LwIP работает с платой расширения USB LAN. По умолчанию драйвер устройства использует одно прерывание для сигнализации как о приеме кадра, так и о завершении передачи. В процессорах ADSP BF533 для этих целей используется прерывание IVG8, а в ADSP BF561 — IVG11. Дополнительную информацию, касающуюся написания драйверов устройств, можно найти в документации [3, 4].

#### Рекомендации к программному и аппаратному обеспечению

##### Рекомендации к программному обеспечению

В реализации для процессоров Blackfin стек LwIP исполняется на платформе VDK. Подразумевается, что пользователи LwIP знакомы с общими концепциями операционных систем реального времени (ОСРВ) [13].

В этом разделе описываются распространенные конфликты программного обеспечения, которые могут возникать при интеграции LwIP с другими программными компонентами или приложениями.

- Любой программный компонент, который использует библиотеки системных служб, потребует, как минимум, применения прерываний и менеджера DMA. При интеграции двух и более проектов, использующих библиотеки системных служб, необходимо, чтобы в системе был только один менеджер прерываний и только один менеджер DMA. Это также распространяется на

менеджеры устройств и отложенных обратных вызовов, а также все прочие службы, используемые в системе.

- Интеграция новых устройств или ресурсов в систему требует дополнительного выделения памяти под менеджер прерываний и менеджер драйверов устройств. Ниже приводится листинг с макросом, используемым для назначения памяти устройствам и каналам DMA. Вы можете подставить свой множитель, исходя из количества каналов и устройств в системе.

```
// Данные менеджера DMA (базовый адрес + память для 2 каналов DMA)
static u8 DMAmgr_storage[ADI_DMA_BASE_MEMORY + (ADI_DMA_CHANNEL_MEMORY * 2)];
// Данные менеджера устройств (базовый адрес + память для 2 устройств)
static u8 devmgr_storage[ADI_DEV_BASE_MEMORY + (ADI_DEV_DEVICE_MEMORY * 2)];
```

Каждый программный компонент может иметь свой собственный набор процедур инициализации, который при интеграции может конфликтовать с процедурами инициализации других компонентов. Стандартные процедуры инициализации включают в себя инициализацию прерываний, инициализацию устройства, настройки сигнала тактовой синхронизации и регулятора напряжения, инициализацию внешней памяти, «кучи» и т. д.

##### Рекомендации к аппаратным средствам

Зачастую работоспособная в автономном режиме система может не работать в связке с другими компонентами из-за ограниченной пропускной способности платформы для встраиваемых приложений. Процессоры Blackfin имеют внутренние шины, разрядность которых для разных представителей различается (в процессорах ADSP BF53x разрядность шин — 16 бит, а в ADSP BF561 — 32 бита). Каждая из внутренних шин процессора может работать с максимальной частотой 133 МГц, и, следовательно, максимальная скорость передачи данных составляет 266 или 532 Мбайт/с, в зависимости от разрядности шины. Существует ряд факторов, которые

могут вызывать уменьшение доступной пропускной способности. К ним относятся задержки переключения системной шины с чтения на запись, конфликты обращений к банкам памяти, задержка из-за арбитража, конфликты между ядром и контроллером DMA и т. д. Эти факторы в различных приложениях проявляются по-разному и зависят от динамического взаимодействия процессов пересылки периферийных данных и обращений ядра. В разделе «Оптимизация системы» обсуждаются методы оптимизации, позволяющие минимизировать влияние факторов, воздействующих на пропускную способность системы, и, как следствие, увеличить эффективность использования системных ресурсов.

#### Управление прерываниями

Одно прерывание IVG может совместно использоваться несколькими периферийными узлами, и, кроме того, несколько программных компонентов могут использовать одно и то же прерывание IVG. При интеграции в таблице векторов событий (EVT, event vector table) должна регистрироваться процедура обслуживания прерывания только для одного из программных компонентов. Другими словами, одному уровню приоритета IVG может быть назначена только одна процедура обслуживания прерывания. Кроме того, назначение периферийным узлам приоритетов и распределение их по разным уровням приоритетов IVG заметно снижает сложность системы.

#### Размещение кода и данных в памяти

При интеграции программных компонентов любая оптимизация карты памяти (например, помещение кода, чувствительного ко времени исполнения, а также кода и данных, к которым наиболее часто обращается приложение, в память L1) потребует проведения ее повторного анализа. Этот момент зачастую игнорируют при использовании программных компонентов от разных производителей, в которых применяется ручная оптимизация карты памяти. При этом поведение приложения (например, процентное соотношение обращений к определенным секциям кода и данных) после интеграции может измениться, что в конечном

```

/* Пользовательский обработчик исключений */
/* Точка входа пользовательского обработчика исключений */
.GLOBAL UserExceptionHandler;
UserExceptionHandler:
/**
 * Обработка исключения...
 */
* Следует помнить, что в VDK пользовательское исключение 0,
* которое вызывается командой EXCPT 0, зарезервировано
*
*
* В этом обработчике вы должны обработать любые другие
* исключения (пользовательские или системные).
*/
RTX;
.UserExceptionHandler.end:

```

Рис. 3. Процедура обслуживания исключений, создаваемая по умолчанию в проекте с поддержкой VDK

итоге приведет к росту задержек обращения к памяти по сравнению с исполнением программных компонентов по отдельности. Поэтому после интеграции следует повторно проанализировать размещение кода и данных в памяти.

#### Обработчики исключений и аппаратных ошибок

Для распределения времени между задачами и управления другими действиями ядра в VDK используется механизм исключений. Обработчик исключения в VDK представляет собой процедуру, интегрированную в код ядра. Эта процедура передает все исключения, обусловленные ошибками, в пользовательскую процедуру обработки исключений, создаваемую по умолчанию. В приложениях, в которых используются только библиотеки системных служб, вам потребуется назначить обработчик исключения вручную. Однако при интеграции приложения, использующего системные службы, в проект, работающий под управлением VDK, выделять отдельный обработчик исключения не следует. В этом случае все исключения необходимо обрабатывать в пользовательской процедуре обслуживания, которая автоматически создается в проекте с поддержкой VDK. Структура пользовательской процедуры обслуживания показана на рис. 3.

#### Прерывания ошибок

Большинство прерываний, вызванных сбоями периферийных узлов, отображено в прерывании IVG7. В связи с этим для эффективной отладки приложения важно локализовать в явном виде источники ошибок DMA или периферийных узлов. Использование в системе прерываний при сбоях — это лучший способ обнаружения переполнения или нехватки данных в том или ином периферийном узле. В качестве примера можно привести ситуацию, когда ресурс, имеющий высокий приоритет, может не давать ресурсу с более низким приоритетом обращаться к памяти, в которой могут содержаться данные или даже дескриптор DMA.

#### Оптимизация системы

В этом разделе описывается несколько методов программной и аппаратной оптимизации, которые помогут разработчикам в пол-

ной мере использовать все преимущества архитектуры процессоров Blackfin.

#### Аппаратная оптимизация

Существует целый ряд методов оптимизации системы. Ниже приводятся рекомендуемый минимум:

- **16/32-разрядные пересылки.** При обмене в режиме DMA с периферийными узлами и в операциях DMA «память–память» всегда используйте максимальную разрядность шины. В процессорах ADSP BF53x должны инициироваться 16-разрядные, а в ADSP BF561 — 32-разрядные пересылки.
- **Эффективное использование каналов DMA.** Не допускайте в одном канале DMA запуска двух пересылок, которые пересекались бы во времени.
- **Разбиение банков SDRAM.** Для одновременного обращения к нескольким буферам данных и минимизации времен переключения задайте для SDRAM конфигурацию с 4 банками. Например, плата ADSP BF561 EZ KIT Lite имеет 64 Мбайта SDRAM, которые можно сконфигурировать как четыре банка по 16 Мбайт. Для эффективного использования структуры с разбиением на банки не размещайте два буфера данных, обращение к которым производится одновременно, в одном банке SDRAM.
- **Управление трафиком DMA.** Для эффективного расходования пропускной способности системы используйте регистры управления трафиком DMA. Эти регистры позволяют влиять на частоту смены направления передачи на шинах данных, автоматически группируя передачи с совпадающим направлением. Более подробную информацию об оптимизации DMA можно найти в руководствах [2, 3 или 4].
- **Сведите к минимуму конфликты между ядром и контроллером DMA.** Активизация кэша данных и команд может существенно увеличить производительность приложения. Если программа имеет очень большой размер (что характерно для приложений LwIP/VDK), кэширование может дать значительный прирост производительности. Более подробная информация о методах оптимизации содержится в литературе [4, 6 и 8].

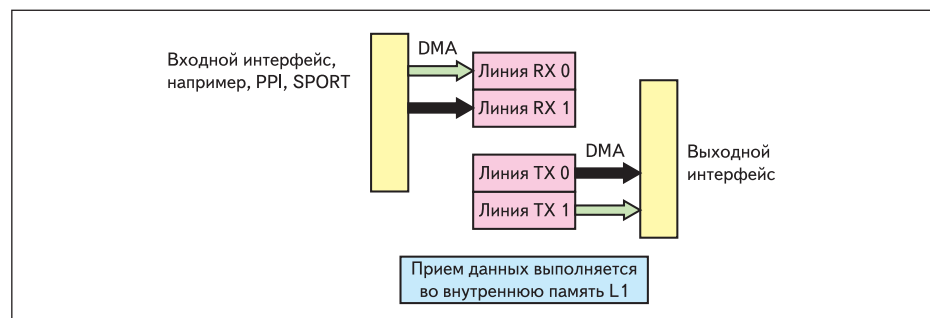


Рис. 4. Обработка данных в простой схеме с двойной буферизацией

### Программная оптимизация Выполнение процедур обслуживания прерывания

Избегайте обработки данных в процедурах обслуживания прерываний и обратных вызовах. Буферы данных следует обрабатывать в пользовательских потоках. Уровень приоритета прерываний выше, чем у определяемых пользователем потоков. Определяемые пользователем потоки всегда выполняются на уровне приоритета прерывания IVG15 (наименьший приоритет), а процедуры обслуживания прерываний и обратные вызовы [7] — на соответствующих им уровнях приоритета. Для более эффективного использования времени CPU понижайте приоритет процедур обслуживания прерываний при помощи отложенных обратных вызовов.

Низкоприоритетная процедура обслуживания прерывания отложит обработку высокоприоритетной процедуры обслуживания прерывания, если первая не поддерживает вложение. В процессорах Blackfin реализован механизм вложения прерываний, который позволяет высокоприоритетным периферийным узлам прерывать выполнение низкоприоритетных прерываний.

### Эффективная схема управления буферами

Во избежание простоев процессора используйте схему двойной буферизации. Применение двойной или множественной буферизации позволит процессору обрабатывать текущий буфер во время заполнения других буферов каналом DMA периферийного узла. На рис. 4 показана простая схема управления данными при двойной буферизации. Дополнительные примеры схем управления двоянными или множественными буферами данных можно найти в книге [8].

### Оптимизация при компиляции

Для повышения производительности приложения может использоваться оптимизация при компиляции. Наилучшие результаты достигаются при использовании ключей компилятора -o и -ira. Дополнительного улучшения производительности можно добиться применением оптимизации по результатам профилирования (PGO, profile-guided optimization). Подробное описание методов оптимизации можно найти в главе руководства по компилятору [14] "Achieving Optimal Performance from C/C++ Sources" (обеспечение максимальной производительности исходного кода на C/C++). Для повышения производительности можно вручную оптимизировать некоторые процедуры обслуживания прерываний и чувствительные ко времени исполнения функции, реализовав их на языке ассемблера.

### Оптимизация при компоновке

Внутренняя память процессора Blackfin представляет собой SRAM первого уровня (SRAM L1), которая зачастую используется неэффективно. Помещение чувствительных ко времени исполнения частей программы

и частей программы, к которым чаще других происходит обращение, в SRAM L1 позволяет избежать большинства потерь циклов, возникающих вследствие промахов кэша. Для локализации наиболее часто исполняемых функций и их размещения в памяти SRAM L1 используйте статистический профайлер (Tools -> Statistical profiler). Кроме того, для обеспечения детерминированного времени реакции в SRAM L1 следует размещать критические процедуры обслуживания прерываний. Подробная информация о размещении кода в памяти в процессорах Blackfin дана в статье [9].

### Общие рекомендации по отладке

В дополнение к стандартным возможностям, таким как программные точки останова и окна вычисления выражений (Expressions) или просмотра содержимого локальных переменных (Locals), средой VisualDSP++ совместно с аппаратными средствами архитектуры Blackfin обеспечивается богатый набор инструментов, позволяющих отлаживать сложные системы. Рассмотрим некоторые из ключевых инструментов отладки.

### Окно истории VDK

Окно истории VDK (рис. 5) используется для наблюдения за состоянием потоков, событиями, конфликтами потоков и т. д. Это окно особенно полезно при отладке многопоточных приложений. Окно истории VDK вызывается из основного меню VisualDSP++ IDDE комбинацией View -> VDK windows -> History.

### Окно состояния VDK

В окне состояния VDK (рис. 6) отображается состояние всех потоков, семафоров, сообщений и других элементов системы. Окно состояния VDK также отображает код ошибки в ситуации «паники ядра» (Kernel Panic). Наиболее распространенными ошибками, порождающими панику ядра, является использование привилегированных (или запрещенных) API-функций VDK в процедурах обслуживания прерываний или обратных вызовах, а также обращение к уже уничтоженным ресурсам потока. Окно состояния VDK вызывается из основного меню VisualDSP++ IDDE комбинацией View -> VDK windows -> Status.

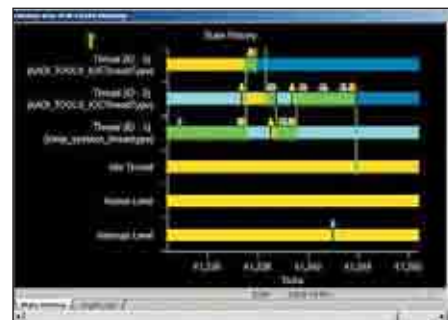


Рис. 5. Окно истории VDK

### Блок трассировки и регистры программного автомата

Для отладки исключения необходимо локализовать его причину, проанализировав содержимое поля EXCAUSE регистра SEQSTAT. Адрес команды, которая вызвала исключение, сохраняется в регистре RETX. При возникновении исключения запустите приложение повторно, поместив при этом точку останова на адрес команды, вызвавшей исключение, и активизировав буфер трассировки установкой двух битов в регистре TBUFCTL. После этого можно выявить точную причину исключения, исследовав последовательность команд, находящихся в буфере трассировки.

### Блок контрольных точек

При помощи регистров «контрольных точек» (watchpoint) можно отлаживать ситуации, когда ошибка возникает только при  $n$ -ом обращении к команде или элементу данных. Регистры контрольных точек позволяют в таких случаях сгенерировать прерывание эмуляции при  $(n-1)$ -ом обращении к команде или элементу данных, после чего можно перейти в режим пошагового исполнения для поиска причины исключения или ошибки.

### Регистры состояния периферийных узлов

При обнаружении ошибки периферийного узла или канала DMA ее причина может быть выявлена с помощью соответствующих регистров состояния. Причиной ошибок могут быть как неправильная конфигурация, так и неэффективное построение системы (например, ошибки переполнения или нехватки данных в каналах DMA). Избавиться от ошибок второго вида поможет более эффективное управление буферами данных или дальнейшее разбиение приложения на задачи более низкого уровня.

При использовании библиотек системных служб (SSL) большинство причин ошибок может быть выявлено по кодам, возвращаемым API функциями системных служб, а также по кодам событий в функциях обратных вызовов. Библиотеки SSL позволяют идентифицировать ошибки в работе периферийных узлов и DMA. В качестве примера может использоваться любой из



Рис. 6. Окно состояния VDK

примеров, входящих в состав Multimedia Starter Kit [14].

В этом разделе были кратко представлены лишь некоторые из доступных методов отладки. Дополнительную информацию можно найти в публикации [17].

### Заключение

Интеграция программных компонентов — это сложная, нетривиальная задача. При объединении нескольких программных компонентов возрастают требования к пропускной способности, усложняется анализ использования системных ресурсов, разрешение потенциальных программных и аппаратных конфликтов, а также отладка приложения. Приведенные в статье рекомендации позволят избежать распространенных ошибок, возникающих при интеграции программных компонентов, и разработать оптимизированное и устойчивое приложение.

### Возможности, поддерживаемые LwIP

Список возможностей LwIP, поддерживаемых на момент написания статьи, приведен в ZIP-архиве, который находится на сайте Analog Devices [21]. Новые возможности,

добавленные в последующих версиях и обновлениях VisualDSP++, перечислены на странице: <http://www.analog.com/processors/blackfin/evaluationDevelopment/crosscore/toolsUpgrades/index.html>. ■

### Литература

1. LwIP User Guide, <каталог\_установки>\Analog Devices\VisualDSP 4.5\Blackfin\lib\src\lwip\docs.
2. BF537EthernetDeviceDriverDesign.doc, <каталог\_установки>\Analog Devices\VisualDSP 4.5\Blackfin\lib\src\lwip\docs.
3. SMSCLAN91C111\_DeviceDriver.doc, <каталог\_установки>\Analog Devices\VisualDSP 4.5\Blackfin\lib\src\lwip\docs.
4. ADSP-BF533 Blackfin Processor Hardware Reference. Rev 3.2, July 2006. Analog Devices, Inc.
5. ADSP-BF561 Blackfin Processor Hardware Reference. Rev 1.1, February 2007. Analog Devices, Inc.
6. ADSP-BF537 Blackfin Processor Hardware Reference. Rev 2.0, December 2005. Analog Devices, Inc.
7. Embedded Media Processing. David Katz and Rick Gentile. Newnes Publishers, Burlington, MA, USA, 2005.
8. Charles Poynton. Digital Video and HDTV. Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 2003.
9. Video Framework Considerations for Image Processing on Blackfin Processors (EE-276). Rev 1, September 2005. Analog Devices, Inc.
10. VisualDSP++ 4.5 Device Drivers and System Services Manual for Blackfin Processors. Rev 2.0, March 2006. Analog Devices, Inc.
11. Video Templates for Developing Multimedia Applications on Blackfin Processors (EE-301). Rev 1, September 2006. Analog Devices, Inc.
12. PGO-Linker- A Code Layout Tool for the Blackfin Processors. (EE-306). Rev 1, December 2006. Analog Devices, Inc.
13. VisualDSP 4.5 Kernel (VDK) Users Guide. Rev 2.0, April 2006. Analog Devices, Inc.
14. Multimedia Starter Kit. Analog Devices Inc. <http://www.analog.com/processors/platforms/msk.html>
15. VisualDSP++ 4.5 Linker and Utilities Manual, Rev 2.0, April 2006. Analog Devices, Inc.
16. VisualDSP++ 4.5 Blackfin C/C++ Compiler and Library Manual, Rev 4, April 2006, Analog Devices, Inc.
17. Blackfin Processor Troubleshooting Tips Using VisualDSP++ Tools (EE-307). Rev 1, December 2006, Analog Devices Inc.
18. W. Richard Stevens. Unix Network Programming, volumes 1–2. Prentice Hall, USA, 1998.
19. Jean J. Labrosse. MicroC/OS-II. CMP Books, San Francisco, CA, USA, 2002.
20. Tanenbaum A. Computer Networks, Prentice Hall, USA, 2002.
21. Прилагающийся ZIP File. [http://www.analog.com/UploadedFiles/Associated\\_Docs/630013030EE312v01\\_c3e19208\\_3ea0\\_4336\\_9080\\_2cfcf7bf724a.zip](http://www.analog.com/UploadedFiles/Associated_Docs/630013030EE312v01_c3e19208_3ea0_4336_9080_2cfcf7bf724a.zip)