ПРОТОТИПИРОВАНИЕ АУДИО АЛГОРИТМОВ НА ПК С ИСПОЛЬЗОВАНИЕМ МОДЕЛЬНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ

МАРК КОРЛЕСС (MARK CORLESS)[1] и
АРВИНД ЭНАНТАН (ARVIND ANANTHAN)[2]

[1] *Ведущий инженер по применению, The MathWorks, Мичиган, США*
Mark.Corless@mathworks.com
[2] *Технический маркетинг, The MathWorks, Массачусетс,США*
Arvind.Ananthan@mathworks.com

Персональные компьютеры (ПК) всё чаще применяются в качестве основного окружения для создания, разработки и симуляции аудио алгоритмов и систем обработки звука с потоковым захватом входов с многоканальных аудио устройств. В этой статье мы показываем, как можно разрабатывать и симулировать аудио алгоритмы в текстовой и графической среде моделирования для ПК. Эти модели связываются с типичными многоканальными аудио устройствами через PortAudio, что позволяет общаться по стандартным аудио интерфейсам, таким, как Direct Sound, WDM-KS и ASIO. Кроме того, в этой статье рассматриваются некоторые типичные сложности, с которыми сталкиваются инженеры при симуляции потокового многоканального аудио алгоритма – такими, как синхронизация каналов, потери пакеты и проблемы с задержками. Мы также демонстрируем технику для минимизации и измерения задержки в обе стороны. Мы использовали три набора аппаратуры для наших экспериментов - Behringer UCA202, M-Audio Delta 66 и M-Audio Firewire 410 – для передачи потокового аудио через модель. В качестве примера в этой статье используется алгоритм автоматического управления коэффициентом усиления, который моделируется с использованием комбинации Simulink, Signal Processing Blockset, Stateflow и кода Embedded MATLAB. Прототипирование в таком окружении позволяет разработчикам исследовать идеи и осуществлять верификацию корректности проекта на ранней стадии разработки, тем самым уменьшая число итераций на более поздних стадиях, когда исправлять проблемы уже дороже.

ВВЕДЕНИЕ

Персональные компьютеры (ПК) получили широкое распространение для разработки и прототипирования аудио алгоритмов. Разработчики используют программные интерфейсы (API), такие, как PortAudio, которые скрывают сложности работы с аудио устройствами и упрощают разработку аудио алгоритмов [1]. Разработчики обычно используют PortAudio для взаимодействия по таким интерфейсам, как Windows Driver Model-Kernel Streaming (WDM-KS) или Audio Streaming Input Output (ASIO) для эффективной передачи синхронизованного, многоканального звука с низкими задержками на ПК [2].

Многие компании, разрабатывающие продукты, связанные с обработкой сигналов, применяют модельно-ориентированное проектирование для идентификации проблем на ранней стадии процесса разработки и уменьшения времени выхода на рынок разрабатываемых встраиваемых систем [3]. Некоторые аспекты модельно-ориентированного проектирования включают в себя создание моделей для симуляции, позволяющие задавать и исследовать функциональное поведение, реализовать спецификации посредством генерации Си кода и непрерывно тестировать и верифицировать проект относительно требований. Применяя модельно-ориентированное проектирование, разработчики часто используют такие инструменты, как MATLAB, Simulink и Stateflow. Примеры применения этих инструментов включают в себя:

- домашние кинотеатры и системы звук-вокруг [4]
- системы подавления эха и шума для систем "без рук" [5]
- системы воспроизведения аудио [6] для автомобилей
- ресиверы цифрового радио (DAB) [7]

Инструменты для модельно-ориентированного проектирования могут использоваться для прототипирования аудио алгоритмов на ПК с применением PortAudio для таких интерфейсов как ASIO. В этой статье мы демонстрируем платформу на базе ПК для прототипирования аудио алгоритмов. Мы рассматриваем моделирование аудио алгоритма, который включает в себя передачу потокового аудио через модель, настройку алгоритма в реальном времени, а затем обсуждаем типичные проблемы – такие, как задержки и потери пакеты. Сравнение задержек осуществляется для стерео аудио карты Behringer UCA202 и многоканальной карты M-Audio Firewire 410.

1 МОДЕЛИРОВАНИЕ АУДИО АЛГОРИТМА

Аудио алгоритмы могут быть представлены разными способами - в зависимости от типа алгоритма, уровня сложности и предполагаемой целевой аудитории. Поток сигналов и поток состояний часто удобней представить в графическом виде, что упрощает представление алгоритма. Текстовые описания часто являются

предпочтительными, если нужны высокие уровни детализации. Смешение графических и текстовых описаний сильно зависит от того, как именно разработчик предпочитает визуализировать идеи и обмениваться ими.

Эффективная платформа для прототипирования должна быть достаточно гибкой, чтобы описывать потоки сигналов и управляющую логику в графическом виде, в то же время позволяя разработчику описывать алгоритм в текстовом виде. Мы продемонстрируем эту гибкость в описании алгоритма на примере автоматического управления коэффициентом усилением (automatic gain control, AGC), используя комбинацию Simulink, Stateflow и кода Embedded MATLAB.

AGC широко применяется в аудио алгоритмах. Например, в цифровых камерах звук записывается одновременно с записью видео. AGC для аудио требуется для усиления речевых сегментов до требуемого уровня, при этом подавляя сегменты, содержащие только шумы [1]. Аналогично, в приложениях Voice over Internet Protocol (VoIP), где уровни аудио могут значительно меняться в зависимости от окружения и пользовательских настроек, AGC используется для подстройки уровня аудио сигнала для достижения равномерного и комфортного уровня [9]. Хотя AGC является распространенным алгоритмическим компонентом, разработчики продолжают развивать фундаментальные техники и экспериментировать с новыми техниками для улучшения работы алгоритма с сохранением эффективности реализации.

1.1 Описание поведения при помощи потоков сигналов

Simulink и Signal Processing Blockset [10] были выбраны в качестве инструмента графического описания потоков сигналов в алгоритме на высоком уровне. Графическое описание потоков сигналов позволяет разработчикам создавать высокоуровневые исполняемые спецификации, похожие на диаграммы, обычно приводящиеся в книгах и статьях. Графическое описание потоков сигналов часто является предпочтительной начальной точкой для высокоуровневого рассмотрения проекта с руководством или несколькими командами. На рис. 1 показано высокоуровневое описание потоков сигналов в Simulink для нашего примера алгоритма AGC; входной сигнал проходит через ряд блоков: оценка уровня (Estimate Level), выделение сигнала (Detect Signal), поддержание уровня (Follow Level) и привязка уровня к усилению (Map Level to Gain) для динамического расчета и применения усиления.

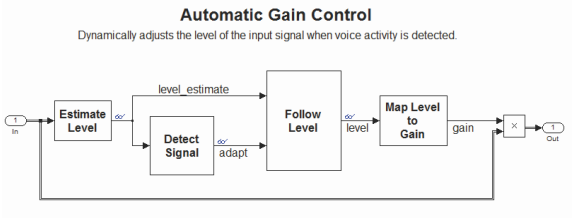

Рис. 1: Поток сигналов в модели AGC.

Подсистема Estimate Level, показанная на рис. 2, также описана с использованием блоков для представления потоков сигналов.

Для оценки уровня используется простая методика выбора максимального абсолютного значения во входном кадре.

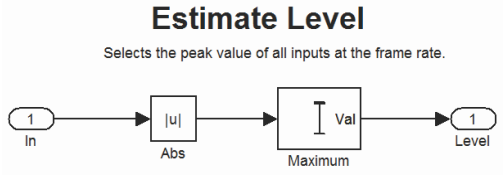

Рис. 2: Поток сигналов в подсистеме Estimate Level.

Подсистема Map Level to Gain, показанная на рис. 3, также описывается с использованием потоков сигналов. Интерполяционная таблица используется для указания формы усиления, а затем усиление применяется для подстройки общего уровня.

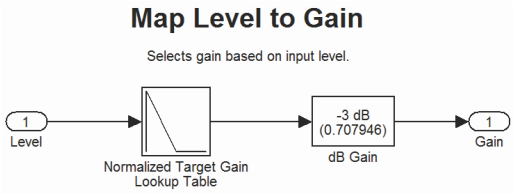

Рис. 3: Описание подсистемы Map Level to Gain.

1.2 Описание поведения с использованием состояний

Некоторые алгоритмы лучше всего представлять с использованием таких конструкций, как состояния и переходы. Stateflow позволяет в графическом виде представить иерархию состояний и параллельные состояния, а также переходы между ними [10]. В этом примере с AGC, подсистема Detect Signal определяет, когда присутствует голосовой сигнал и выдает адаптационный сигнал, показывающий, нужно ли обновлять расчет усиления. Поскольку простой механизм детектирования может быть представлен в виде двух состояний (сигнал детектирован и сигнал не детектирован), мы описали это поведение в Stateflow, как показано на рис. 4.

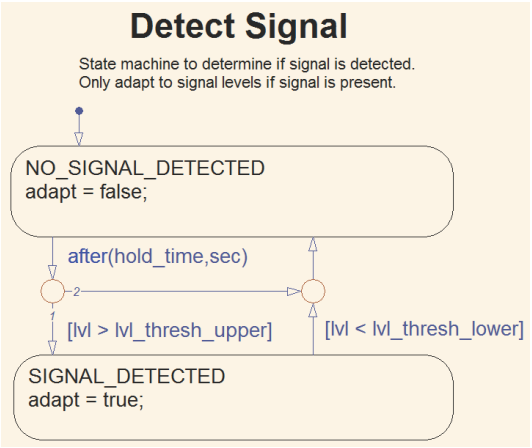

Рис. 4: Описание состояний в подсистеме Detect Signal.

Детектор перемещается между состояниями в зависимости от уровня входного сигнала с учетом верхнего и нижнего порогов. Дополнительный гистерезис достигается за счет ключевого слова `after`, что позволяет указать, сколько секунд надо находиться в состоянии "сигнал не детектирован" до перехода в состояние "сигнал детектирован".

1.3 Описание поведения с использованием текстового языка программирования

Зачастую разработчикам аудио алгоритмов привычно программировать с использованием текстовых языков, таких, как Си и MATLAB. Описание потоков управления, таких, как вложенные конструкции if-then-else, может быть осуществлено графически, но разработчик может предпочесть текстовое описание. Подсистема Level Follower является примером простого вложенного потока if-then-else, используемого для отслеживания пиков оцениваемого уровня сигнала, основанного на адаптационном сигнале. Мы выбрали код Embedded MATLAB [10] для описания подсистемы Level Follower, как показано на рис. 5. Embedded MATLAB, будучи подмножеством языка MATLAB, поддерживает ускорение симуляции, а также генерацию Си кода.

```
function level  = follow_level(level_estimate,...
                                adapt, atk, dcy)
% atk = Attack rate, typically close to 1
% dcy = Decay rate, typically much smaller than atk
persistent level_n1
if isempty(level_n1)
    level_n1 = 0;
end

if adapt
    err = level_estimate - level_n1;
    if err > 0
        atk_dcy = atk;   % Attack (quickly)
    else
        atk_dcy = dcy;   % Decay (slowly)
    end
    level = level_n1 + err * atk_dcy;
else
    level = level_n1;
end

level_n1 = level;
```

Рис. 5: Текстовое описание подсистемы Level Follower.

Simulink, Stateflow и Embedded MATLAB поддерживают интеграцию внешнего Си кода в среду симуляции; эта поддержка не демонстрируется в этом примере. Интеграция существующего кода позволяет разработчикам повторно использовать существующую интеллектуальную собственность и библиотеки во время разработки, а также интегрировать оптимизированные под железо функции.

1.4 Оценка поведения

Для оценки поведения алгоритма AGC, мы создали тестовую модель, показанную на рис. 6. В качестве входа в тестовой модели используется звуковой Wave файл, который проходит через AGC и результаты симуляции записываются.

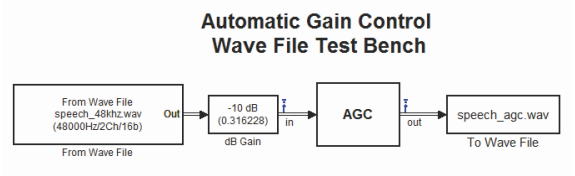

Рис. 6: Тестовая модель AGC.

Сигналы *level_estimate*, *level* и *adapt* в алгоритме AGC выводятся на осциллограф Simulink, который представлен как иконка с очками рядом с соответствующими сигналами на рис. 1. На рис. 7 показан график осциллографа Simulink. Мы использовали эту визуализацию, чтобы убедиться, что алгоритм функционирует корректно.

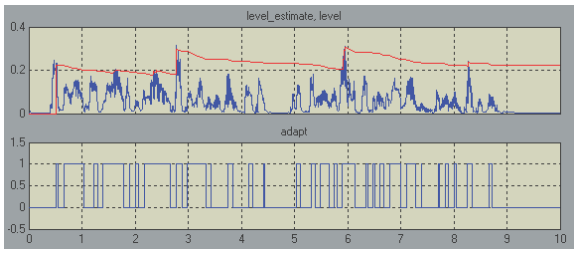

Рис. 7: Промежуточные сигналы AGC на осциллографе.

Мы также использовали MATLAB для пост-обработки результатов симуляции. Например, сигналы *in* и *out* тестовой модели были записаны в рабочую область MATLAB, на что указывают иконки с антенной рядом с соответствующими сигналами на рис. 6. Затем мы написали MATLAB скрипт для построения графика и форматирования результатов, как показано на рис. 8.

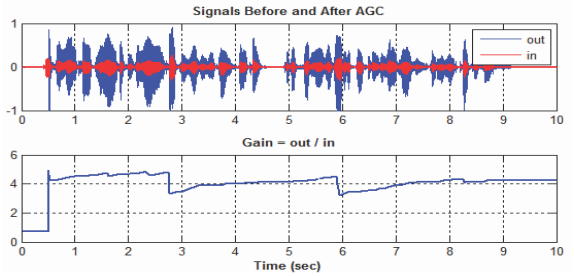

Рис. 8: График MATLAB для записанных сигналов.

С использованием такой платформы, разработчики могут описывать свои идеи различными способами, включая потоки сигналов, машины состояний и текстовые описания. Симуляция позволяет ясно понять поведение алгоритма, позволяя разработчику отслеживать внутренние сигналы, а также записывать сигналы для пост-обработки. Эти результаты могут быть использованы в качестве базы для изучения возможных улучшений в алгоритме, таких, как более умные способы детектирования голосового сигнала или эффекты от разных профилей усиления.

2 ПОТОКОВОЕ АУДИО В МОДЕЛИ

Когда базовая структура алгоритма становится понятной посредством симуляции, часто возникает необходимость динамически настраивать и оценивать влияние изменений параметров на качество звука в реальном времени. Эта платформа использует блоки From Audio Device (со звукового устройства) и To Audio Device (на звуковое устройство), которые предоставляются в Signal Processing Blockset, для поддержки потоковой передачи аудио через звуковые устройства. Совместно, блоки Audio Device позволяют разработчикам передавать потоковое аудио через окружение Simulink для прототипирования аудио алгоритмов.

В следующих разделах мы предлагаем обзор подключения аудио устройств и примеры передачи потокового аудио через модель, с обеспечением многоканальной синхронизации и настройки параметров модели во время симуляции в реальном времени.

2.1 Взаимодействие с аудиоустройствами

Для разработки аудио алгоритмов на ПК разработчики могут выбирать из широкого набора размеров, типов и форматов аудио устройств. Разработчики выбирают аудио устройство исходя из частоты дискретизации (например, 44.1 кГц, 48 кГц, или 96 кГц), типа сигнала (например, аналоговые, S/PDIF или ADAT), а также выбирают из широкого набора интерфейсов ПК (например, PCI/PCMCIA, USB или Firewire) [11].

Большинство аудио устройств предоставляют интерфейс DirectSound; более производительные аудио устройства предоставляют интерфейсы ASIO и/или WDM-KS [12]. Ключевое различие между этими интерфейсами заключается в задержках. ASIO обычно обладает меньшими задержками, у WDM-KS часто сравнимые задержки с ASIO, и у DirectSound самые большие задержки.

API PortAudio позволяет разработчикам получить доступ к этому широкому набору аудио устройств используя стандартный интерфейс. Поскольку блоки Audio Device используют PortAudio для взаимодействия с аудио устройствами, они могут поддерживать широкий набор устройств. По умолчанию, блоки Audio Device используют PortAudio, скомпилированный с поддержкой интерфейса DirectSound, поскольку почти все аудио устройства для ПК поддерживают этот интерфейс. Для получения преимуществ и снижения задержек при использовании WDM-KS или ASIO, разработчики могут перекомпилировать PortAudio для получения доступа к этим и другим интерфейсам [16].

Мы использовали три аудио устройства в наших экспериментах для передачи потокового аудио через модель при симуляции: Behringer UCA202 [13], M-Audio Delta 66 [14] и Firewire 410 [15]. Мы исследовали конфигурацию PortAudio по умолчанию с поддержкой DirectSound, а также перекомпилировали PortAudio с поддержкой ASIO и WDM-KS.

2.2 Передача потокового аудио из модели

Мы создали модель тестовой обвязки, в которой происходит чтение из Wave файла и передача потокового аудио с использованием блока To Audio Device, как показано на рис. 9. Мы также добавили блок Manual Switch, который позволяет вручную переключаться между оригинальным источником звука и выходом алгоритма AGC. Такая конфигурация позволила нам прослушивать выходной сигнал, при этом продолжая наблюдать за активностью внутренних сигналов на осциллографе Simulink.

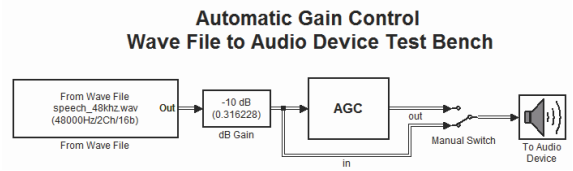

Рис. 9: Тестовая модель для алгоритма AGC.

Когда мы были удовлетворены результатами тестирования относительно входного Wave файла, мы заменили блок From Wave File на блок From Audio Device, как показано на рис. 10. Такая конфигурация позволяла нам осуществлять верификацию относительно внешнего источника аудио.

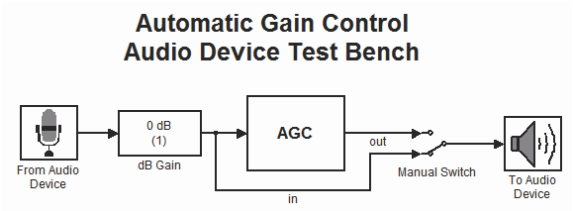

Рис. 10: Тестирование алгоритма AGC с внешним источником аудио.

2.3 Обеспечение многоканальной синхронизации

В предыдущих тестах как входы, так и выходы, были стерео сигналами. При работе с дополнительными выходными каналами, PortAudio предоставляет блок Audio Device с выбором конфигурации. Можно выбрать стерео пары, или передачу всех сигналов через один порт. Например, M-Audio Firewire 410 предоставляет опции для выбора индивидуальных пар (например, Line 1/2, Line 3/4, Line 5/6 и Line 7/8), а также опцию *Multi* для всех выходов.

Разработчик может использовать несколько блоков To Audio Device в модели для направления разных стерео каналов на разные каналы (Line) аудио устройства. Такой подход хорош, если эти каналы со стерео данными не связаны с друг с другом; если же они связаны, как в системе Dolby 5.1, то сигналы могут быть направлены на выход с использованием опции *Multi*, учитывающей то, как Windows работает с драйверами устройств. Если каналы заданы с использованием нескольких блоков To Audio Device, то дополнительные задержки будут вноситься между каналами на уровне драйвера устройства [17]. Такая дополнительная задержка между каналами может привести к нежелательному сдвигу звуковой картинки [20].

Пример конструкции для моделирования синхронизации выходных каналов приведен на рис. 11. В этом примере вход является двухканальным сигналом с размером пакета в 128 сэмплов. Сигнал передается в блок Matrix Concatenation, который упаковывает сигналы в 8 каналов данных, которые будут отправлены на аудио устройство. Такая конструкция обеспечивает синхронизацию всех 8 выходных каналов.

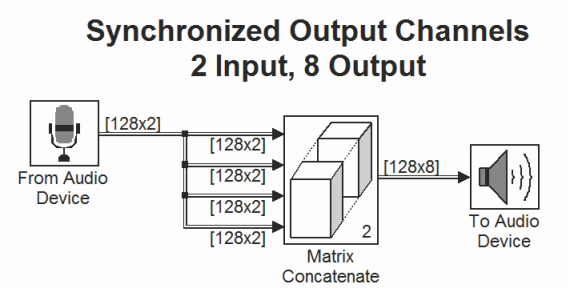

Рис. 11: Конструкция для синхронизации выходных каналов.

3 НАСТРОЙКА ПАРАМЕТРОВ В РЕАЛЬНОМ ВРЕМЕНИ

Зачастую основной мотивацией для передачи потокового аудио через модель является необходимость интерактивной настройки параметров алгоритма в реальном времени. В этом разделе мы обсуждаем графические и программные способы взаимодействия с симуляцией во время обработки аудио.

3.1 Изменение параметров в диалоговых окнах блоков

Многие параметры в блоках Simulink являются настраиваемыми. Настраиваемый параметр – это параметр, значение которого можно менять без повторной компиляции модели [18]. Например, параметр Gain блока dB Gain, показанного на рис. 12, является настраиваемым. Значение Gain может быть изменено во время симуляции. Если параметр не является настраиваемым, то во время симуляции диалоговое окно для этого параметра не позволяет его менять.

Во время тестирования алгоритма AGC, мы настраивали блок dB Gain подсистемы Map Level to Gain и слушали, как это отражается на амплитуде.

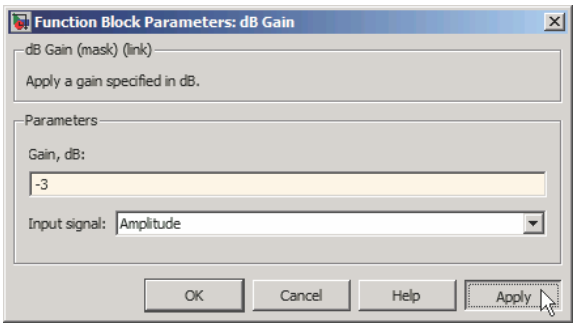

Рис. 12: Диалоговое окно блока dB Gain

3.2 Изменение параметров с использованием переменных MATLAB

Зачастую удобно представлять параметры алгоритма в виде переменных MATLAB. Например, подсистема Detect Signal задает время задержки при переходе от не детектированного к детектированному сигналу в виде переменной `hold_time`. Во время симуляции `hold_time` может быть программно изменена в командном окне MATLAB, как показано на рис. 13. Это изменение можно применить к модели Simulink, нажав кнопку Update Diagram в панели инструментов Simulink.

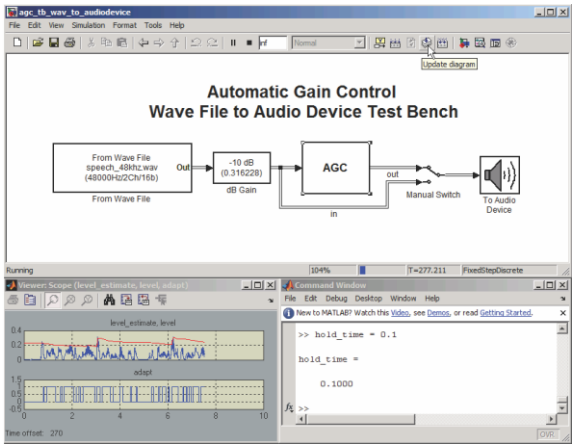

Рис. 13: Настройка переменной `hold_time`.

Такой программный подход может быть расширен для синхронизации изменений нескольких параметров. Например, на рис. 14 показан скрипт MATLAB, который мы использовали для настройки параметров в подсистемах Detect Signal и Follow Level алгоритма AGC. Последняя функция `set_param` в скрипте предназначена для того, чтобы программно обновить модель Simulink. Это действие эквивалентно нажатию кнопки Update Diagram.

```
%% Specify Parameters
hold_time = 0.6; % Signal Detector: Hold time (seconds)
atk = 0.91;      % Follow Level:    Attack coefficient
dcy = 0.03;      % Follow Level:    Decay coefficient

%% Update Simulink Diagram with Changes
set_param('agc_tb_wav_to_audiodevice',...
          'SimulationCommand','update')
```

Рис. 14: Скрипт для программного обновления параметров модели.

В этом разделе мы показали базовые техники для изменения параметров во время симуляции. Разработчики могут использовать эти техники для создания дополнительных интерфейсов по настройке параметров. Например, разработчики могут создать собственный интерфейс пользователя в MATLAB, позволяющий взаимодействовать с моделью во время симуляции [19].

4 ПРЕДОТВРАЩЕНИЕ ПОТЕРИ ПАКЕТОВ

Для нашего простого алгоритма AGC мы смогли успешно передавать потоковое аудио в конфигурации системы по умолчанию без потери пакетов данных. По мере возрастания размера и сложности алгоритма, возможно, что ПК не будет справляться с чтением и записью в буферы аудио устройства, что приведет к потере пакетов. Потерянные пакеты проявляют себя в виде щелчков, хлопков или прерывистого аудио. В этом разделе мы обсудим некоторые основные концепции, которые разработчики должны принимать во внимание для улучшения пропускной способности системы и предотвращения потери пакетов.

4.1 Управление процессами Windows

Обычно Simulink это лишь один из многих процессов, запущенных в операционной системе Windows. Эти процессы разделяют общее время ЦПУ. Разработчики аудио алгоритмов на ПК часто отключают фоновые приложения (например, резервное копирование или антивирус) и отключают сетевые соединения [21]. Мы использовали диспетчер задач Windows для оценки влияния других процессов на использование ЦПУ. Поскольку влияние других процессов может быть случайным, мы оставили только те процессы, которые нужны для разработки, и отключили такие сервисы как сетевые подключения во время демонстрации более сложных алгоритмов.

4.2 Улучшение скорости симуляции в Simulink

При моделировании алгоритмов в Simulink, разработчики должны принимать во внимание некоторые распространенные практики для увеличения скорости симуляции, улучшающие также пропускную способность. Техники для улучшения скорости симуляцию включают пакетную обработку, использование кратных частот дискретизации в многоскоростных моделях и минимизации визуализаций [21][23][24].

В наших тестах мы увидели сильное сокращение использования процессора при уменьшении частоты обновления визуализаций. По умолчанию, блоки для визуализации, такие, как Spectrum Scope и блоки Display, настроены на обновление с той же частотой, что и сигнал, входящий в этот блок. Большинство блоков для визуализации позволяют уменьшать частоту обновления, тем самым сокращая общее время, требуемое на отрисовку визуализации. Мы обнаружили, что более низкая частота обновления оказалась достаточной, чтобы получить представление о работе алгоритма во время потоковой передачи аудио. Пример модели, содержащей блоки Spectrum Scope и Display, настроенные на обновление с меньшей частотой, чем частота сигнала, входящего в этот блок, приводится на рис. 15. Частота обновления блока Display управляется параметром Decimation в диалоговом окне блока. Частота обновления блока Spectrum Scope зависит от его параметра Buffer size. Увеличение Buffer size уменьшает частоту обновления и повышает разрешение визуализации. В этом примере параметр Buffer size установлен равным 4096, так что он обновляется в 16 раз медленней, чем входной сигнал, содержащий пакеты по 256 сэмплов.

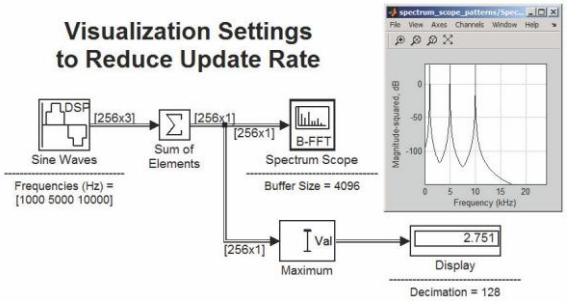

Рис. 15: Уменьшение частоты обновления визуализации.

4.3 Настройка буферов аудио устройств

Буферизация аудио данных является распространенной техникой поддержания непрерывности аудио потока в аудио алгоритмах на ПК [12]. Увеличение размеров этих буферов решает большинство проблем, связанных с щелчками и хлопками, но обратная сторона медали заключается во внесении дополнительной задержки. Кроме того, все изменения, которые пользователь вносит во время работы алгоритма, становятся активными только после начала обработки следующего буфера.

Блоки Audio Device в Simulink позволяют разработчику указать размер буфера, используемого для взаимодействия с аудио устройством, а также длительность очереди, которая находится между блоком Audio Device и аудио устройством. Увеличение размера этих буферов позволяет блокам скомпенсировать скачки в производительности ЦПУ. В наших экспериментах, размер буфера по умолчанию был достаточен для настройки алгоритмов в реальном времени без потери пакетов.

5 СОКРАЩЕНИЕ ЗАДЕРЖЕК

Допустимое значение задержки сильно зависит от приложения, для которого разрабатывается алгоритм. Например, допустимая задержка между аудио и видео может быть до 45 мс [25], в то время как допустимая задержка для мониторинга аудио во время генерации аудио варьируется от 3 до 23 мс [12]. Другая форма задержки – это время, которое проходит между изменением параметра и восприятием этого изменения. В этом случае 100 мс обычно является допустимой задержкой [12].

В этом разделе мы обсуждаем некоторые методики минимизации задержки в обе стороны при прототипирования аудио алгоритмов. Мы также обсуждаем, как настроить аудио устройство в Windows, а также блоки Audio Device в модели. Затем мы обсуждаем метод измерения задержки и предлагаем таблицу измеренных задержек для разных комбинаций ПК и аудио устройств.

5.1 Настройка звукового устройства в Windows

Поскольку драйвер звукового устройства играет важную роль в сокращении задержек, мы исследовали производительность аудио устройства независимо в среде Windows, прежде чем пытаться измерить задержки в окружении Simulink.

Обычно высокопроизводительные аудио устройства предоставляют панель управления для настройки устройства. Например, Behringer UCA202 предоставляет панель управления ASIO-USB, содержащую специальное поле производительности системы. Это поле позволяет выбрать из таких режимов, как Highspeed, Rapid, Fast и Normal. Каждый из этих режимов влияет на ожидаемые задержки.

Чтобы помочь с идентификацией наиболее подходящего режима для наших тестов, мы использовали инструмент CEntrance ASIO Latency Test Utility [26] для измерения задержек в разных конфигурациях. Например, на одном из ноутбуков мы выбрали режим Rapid, поскольку он обеспечивал повторяемость измеряемых задержек в 12,65 мс для размера пакета в 128 сэмплов. Таким образом мы определили, что это наилучшая задержка, которую можно достичь на данном ноутбуке.

5.2 Настройка блоков Audio Device для минимальной задержки

Как описывалось ранее, блоки Audio Device реализованы с использованием нескольких буферов, что помогает предотвратить потерю пакетов при потоковой передаче аудио. Мы выяснили, что при использовании интерфейса DirectSound для потоковой передачи аудио требуются буферы. Однако когда мы переключались на интерфейсы WDM-KS или ASIO, то могли значительно сократить размер этих буферов, что сокращало общую задержку.

Пример конфигурации блока From Audio Device для карты Behringer UCA202 приведен на рис. 16. Чтобы достичь минимальной задержки, мы выставили Queue duration в ноль, а параметры Buffer size и Frame size в минимальную степень двойки, позволяющую предотвратить потерю пакетов. Для нашего теста с использованием Behringer USB 202, мы установили параметры Buffer size и Frame size parameters в 128, что также представляет самое низкое значение, которое можно было использовать в приложении CEntrance для измерения задержек без потери пакетов. Также важно, что для запуска этих тестов мы использовали релиз MATLAB R2009a и соответствующие тулбоксы. С предыдущими версиями MATLAB, возможно, потребуется выставить Queue duration больше, чем в ноль.

Для минимизации времени расчета в блоке Audio Device, параметр Device data type должен быть выбран таким же, как тип данных самого устройства, и параметр Output data type должен быть наименьшим целочисленным типом данных, способным представить тип данных с устройства.

Похожим образом мы настроили блок To Audio Device.

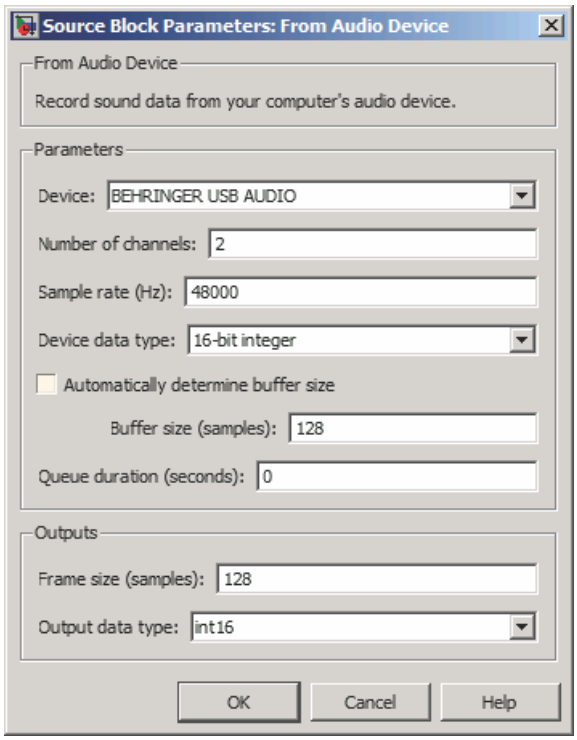

Рис. 16: Параметры блока From Audio Device.

5.3 Измерение задержки

Для изучения достижимых задержек, мы физически подключили выход аудио устройства к его входу. Затем мы создали модель Simulink, которая передавала аудио поток с частотой дискретизации 48 кГц через аудио устройство, как показано на рис. 17. Мы создали тестовую входную переменную в рабочей области MATLAB, состоящую из синусоиды небольшой амплитуды на частоте 6 кГц, которая суммируется с четырьмя периодами большой синусоиды на 1,5 кГц. Подача такого сигнала упростила верификацию того, что никакие пакеты не потеряны и позволила посчитать задержку в обе стороны между входом и выходом.

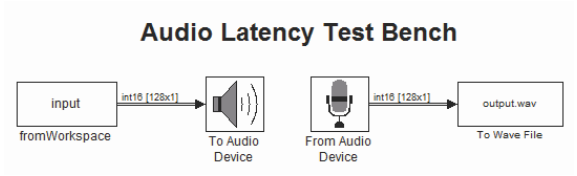

Рис. 17: Передача аудио 48 кГц через аудио устройство.

После запуска теста, мы создали скрипт MATLAB для расчета задержки в обе стороны с использованием взаимозависимости входа и выхода. Выдержка из этого скрипта показана на рис. 18.

```
in   = double(input);
out  = double(wavread('output.wav', 'native'));
c    = xcorr(out, in);
index_center  = (length(c)+1)./2;
[C,index_max] = max(c);
delay         = index_max - index_center;
```

Рис. 18: Скрипт для расчета задержки в обе стороны.

Мы также использовали MATLAB для построения графиков с результатами, чтобы можно было визуально проверить, что никакие пакеты данных не были потеряны во время теста. Пример графиков с результатами тестирования карты M-Audio Delta 66 показан на рис. 19.

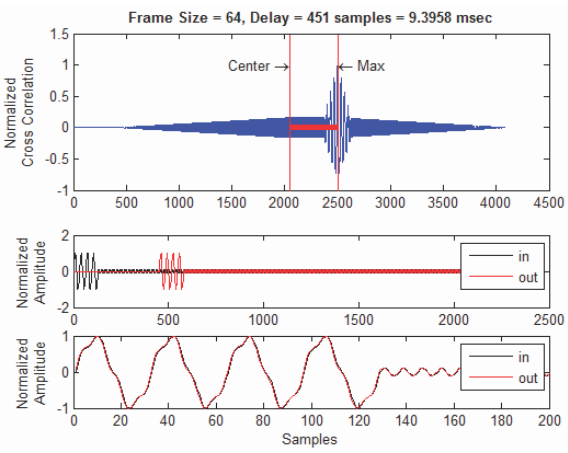

Рис 19: Графики с результатами для карты M-Audio Delta 66 с использованием WDM-KS.

Мы повторили этот тест для нескольких различных комбинаций ПК и аудио устройств. Результаты этих тестов приведены в таблице 1, “Сравнение измеренной задержки в обе стороны на частоте 48 кГц”. В целом, для ASIO, задержка была примерно на 2 пакета больше, чем задержка, измеренная при помощи утилиты CEntrance. Мы связываем это с буферами, требуемыми для блоков Audio Device для взаимодействия с аудио устройством. Наши тесты также показали, что использование 32-битного ПК под управлением Windows XP позволило нам исследовать различные размеры буфера, а при тестировании карты M-Audio Firewire 410 на 64-битном ПК под управлением Windows Vista, размер буфера был зафиксирован в 256 бит. Поскольку 256 бит был также единственным возможным размером буфера в утилите CEntrance, это значение, похоже, является текущим ограничением операционной системы Windows Vista.

Мы выяснили, что такая платформа способна достигать задержки в обе стороны равной 6,73 мс. Такая производительность подходит для разработки многих аудио алгоритмов. Эта платформа также может быть расширена для поддержки автоматической генерации кода для развертывания на распространенных процессорах. Для приложений, требующих еще более низких задержек, разработчики могут воспользоваться преимуществами технологии генерации кода, позволяющей разворачивать модели на встраиваемых цифровых сигнальных процессорах (DSP) [27][28][29].

Таблица 1: Сравнение измеренной задержки в обе стороны на частоте 48 кГц

Аудио устройство	Интерфейс	Буфер	Задержка	
			Пакеты	мс
Behringer UCA 202[1]	ASIO	128	786	16,375
M-Audio Delta-66[2]	ASIO	64	323	6,729
	WDM-KS	64	451	9,396
M-Audio Firewire 410	ASIO	256	1208	25,167
	WDM-KS	256	1472	30,667

[1] Тест на ноубуке Lenovo T60, 32-bit, Windows XP SP3, Intel Core Duo T2500 @ 2GHz, 2GB RAM
[2] Тест на ПК, Windows XP SP3, 32-bit, Intel Core 2 6700 @ 2.67GHz, 3.25GB RAM
[3] Тест на ПК, Windows Vista, 64-bit

6 ЗАКЛЮЧЕНИЕ

В этой статье мы продемонстрировали платформу для прототипирования аудио алгоритмов на ПК в реальном времени.

Эта платформа позволяет разработчикам описывать алгоритмы с использованием графического потока сигналов, состояний и текстовых описаний. В качестве примера мы рассмотрели модель алгоритма автоматического управления коэффициентом усиления, реализованную с использованием Simulink, Signal Processing Blockset, Stateflow и кода Embedded MATLAB. Мы также создали модель тестовой обвязки для анализа и визуализации поведения алгоритма.

Эта платформа позволяет разработчикам передавать потоковое аудио и интерактивно настраивать алгоритм в реальном времени. Для демонстрации этих возможностей мы использовали файл Wave как входной источник в модель автоматического управления коэффициентом усиления и передавали потоковое аудио на выходное устройство. Мы продолжали расширять модель для передачи как входов, так и выходов через аудио устройство. В обоих случаях мы интерактивно настраивали параметры алгоритма в реальном времени и наблюдали за реакцией системы.

Чтобы эффективно применять такой подход для прототипирования аудио алгоритмов, разработчик должен оценить влияние задержек. Мы описали технику для измерения задержки в обе стороны и сравнили измеренные задержки для нескольких разных ПК и аудио устройств. Эта платформа использует PortAudio в качестве интерфейса к аудио

устройству, поэтому является расширяемой для поддержки стандартов и интерфейсов, принятых в индустрии – таких, как DirectSound, WDM-KS и ASIO.

Прототипирование в этом окружении позволяет разработчикам осуществлять верификацию корректности алгоритма на ранних стадиях, тем самым сокращая число итераций и общее время разработки. Хотя это и не обсуждалось детально в данной статье, такая платформа также предлагает возможности для автоматической генерации кода и развертыванию на ПК, а также встраиваемых процессорах.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

[1] Bencina, R. & Burk, P., "PortAudio - an Open Source Cross Platform Audio API," Proceedings of the International Computer Music Conference, Havana, International Computer Music Association, pp.263-266., 2001.

[2] Letz, Stephane, "Porting PortAudio API on ASIO," GRAME - Computer Music Research Lab Technical Note - 01-11-06, November 2001.

[3] Corless, Mark, and Ananthan, Arvind, "Model-Based Design of Fixed-Point Filters for Embedded Systems," Society of Automotive Engineers World Congress 2009-01-0150, April 2009.

[4] Philips Consumer Lifestyle (Philips) Develops One-Piece Surround Sound System with MathWorks Tools, The MathWorks, April 2008: www.mathworks.com/company/user_stories/userstory17418.html

[5] Eslinger, Greg and Dixon, John, "Dynamic Adjustment of System Parameters Improves Echo and Noise Cancellation," Texas Instruments, December 2006: http://focus.ti.com/lit/wp/spry094/spry094.pdf

[6] Schubert, Peter J., Vitkin, Lev, and Braun, David, "Model-Based Development for Event-Driven Applications using MATLAB: Audio Playback Case Study," Systems Engineering, January 2007: http://delphi.com/pdf/techpapers/2007-01-0783.pdf

[7] Harman Becker Designs and Verifies OFDM Radio Receivers Using MathWorks Tools, The MathWorks, March 2009: http://www.mathworks.com/company/user_stories/userstory19636.html

[8] Archibald, Fitzgerald J., "Software Implementation of Automatic Gain Controller for Speech Signal," Texas Instruments SPRAAL1, July 2008.

[9] GIPS Automatic Gain Control, Global IP Solutions, http://www.softfront.co.jp/products/lib/SP_AGC.pdf

[10] http://www.mathworks.com

[11] Walker, Martin, "Choosing A PC Audio Interface: The SOS Guide," Sound On Sound, November 2004: http://www.soundonsound.com/sos/nov04/articles/pcmusician.htm

[12] Walker, Martin, "Optimising The Latency Of Your PC Audio Interface," Sound on Sound, January 2005: http://www.soundonsound.com/sos/jan05/articles/pcmusician.htm

[13] http://www.behringer.com/EN/Products/UCA202.aspx

[14] http://www.m-audio.com/products/en_us/Delta66.html

[15] http://www.m-audio.com/products/en_us/FireWire410.html

[16] Rogers, Alec, "Customized Audio Driver Support (Using ASIO, ALSA, etc. with Signal Processing Blockset Audio Device Blocks)," February 2008: www.mathworks.com/matlabcentral/fileexchange/18599

[17] Schillebeeckx, P., Paterson-Stephens, I. and Wiggins, B., "Using MATLAB/Simulink as an implementation tool for Multichannel Surround Sound," Proceedings of the 19th International AES conference on Surround Sound, Germany, pp. 366-372, June 2001.

[18] Simulink, Tunable Parameters, The MathWorks: http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/index.html?/access/helpdesk/help/toolbox/simulink/ug/f7-20739.html#f7-23615

[19] Fillyaw, Chris, Friedman, Jonathan, and Prabhu, Sameer M., "Creating Human Machine Interface (HMI) Based Tests within Model-Based

Design," Society of Automotive Engineers 2007-01-0780, January 2007.

[20] Fonseca, Nuno and Monteiro, Edmundo, "Latency Issues in Audio Networks," Audio Engineering Society 118th Convention, May 2005.

[21] Burk, Phil, "Audio Latency," 2002: http://www.portaudio.com/docs/latency.html

[22] Simulink, Improving Simulation Performance and Accuracy, The MathWorks: http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/index.html?/access/helpdesk/help/toolbox/simulink/ug/f11-33464.html

[23] Simulink, Performance & Memory Management Guide, The MathWorks: http://www.mathworks.com/support/tech-notes/1800/1806.html

[24] Recorded Webinar: Tips for Speeding up Simulink Models for Signal Processing & Communications, The MathWorks, February 2005: http://www.mathworks.com/company/events/webinars/wbnr30446.html

[25] ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations, Advanced Television Systems Committee, June 2003.

[26] http://www.centrance.com/products/ltu/

[27] Wilber, Jack, "The Sound of Innovation at Cochlear Limited," The MathWorks News & Notes, October 2006: http://www.mathworks.com/company/newsletters/news_notes/oct06/cochlear.html

[28] Ananthan, Arvind, "Rapid-prototyping and Implementing Audio Algorithms on DSPs using Model Based Design and Automatic Code Generation," Audio Engineering Society 123rd Convention, October 2007.

[29] Ananthan, Arvind, "Integrating Simulink Model with VisualDSP++ Project," November 2008: www.mathworks.com/matlabcentral/fileexchange/22243