Solving Audio Subsystem Challenges Using a Crossfade/Level Controller Algorithm in Portable Media Applications

Danny Jochelson
DSP Solutions R&D – Digital Audio Branch
Texas Instruments, dsjochel@ti.com

Jason Kridner
Imaging and Audio Group
Texas Instruments, jdk@ti.com

Agenda

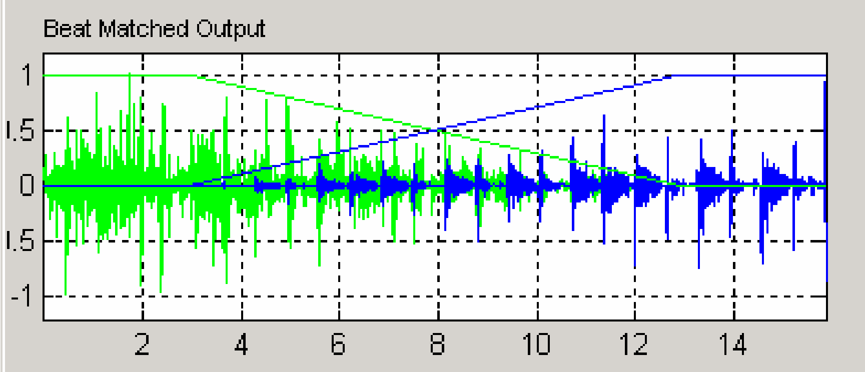
- Motivation
- Common Digital System Architectures and Challenges
- Crossfade/Level Controller Features
  - Stream Relationships
  - Construction of Volume Fades
  - Real-time event functionality
- Implementation for Common Scenario
- Results
- Demonstrations

Level Control Motivation

- More multimedia capabilities on portable devices yields more simultaneous audio streams
  - Standard media
    - Music (including multi-stream mixing, e.g. music with MIDI background track)
    - Audio with video
  - User alerts
    - User Interface sounds
    - Cellular ringtones (MIDI, MP3, etc.)
  - Communication streams
    - Voice streams

Crossfade Motivation

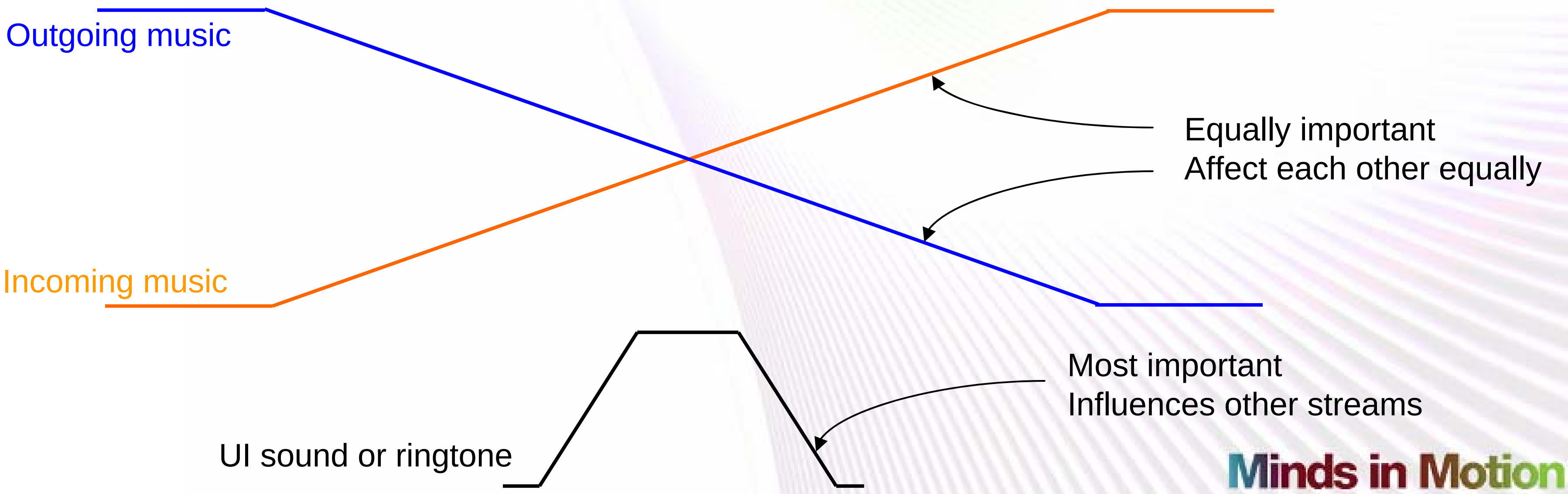
- Crossfades often employed in music clubs for smooth transition between songs
- Can also perform beat-matched crossfade for even smoother transition



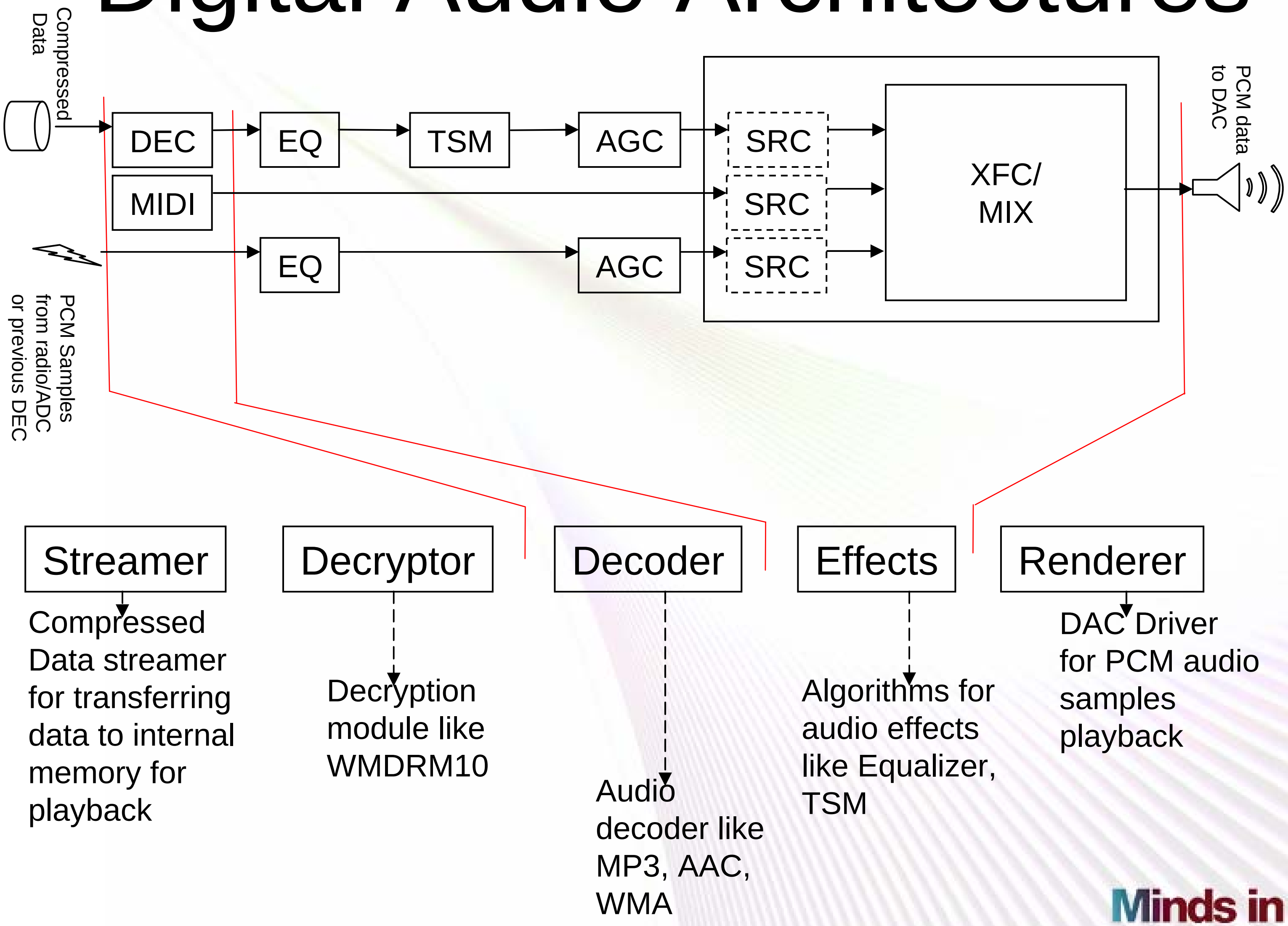
- Feature not commonly found on consumer media devices (e.g. cell phones, portable media players)
  - Can greatly improve user experience (i.e. perception of quality)

Digital Audio Common Use Case

- Crossfade with UI/ringtone (alert) sound
- Establish stream relationships
- Stream relationships change over time



Digital Audio Architectures



Streamer: Compressed Data streamer for transferring data to internal memory for playback

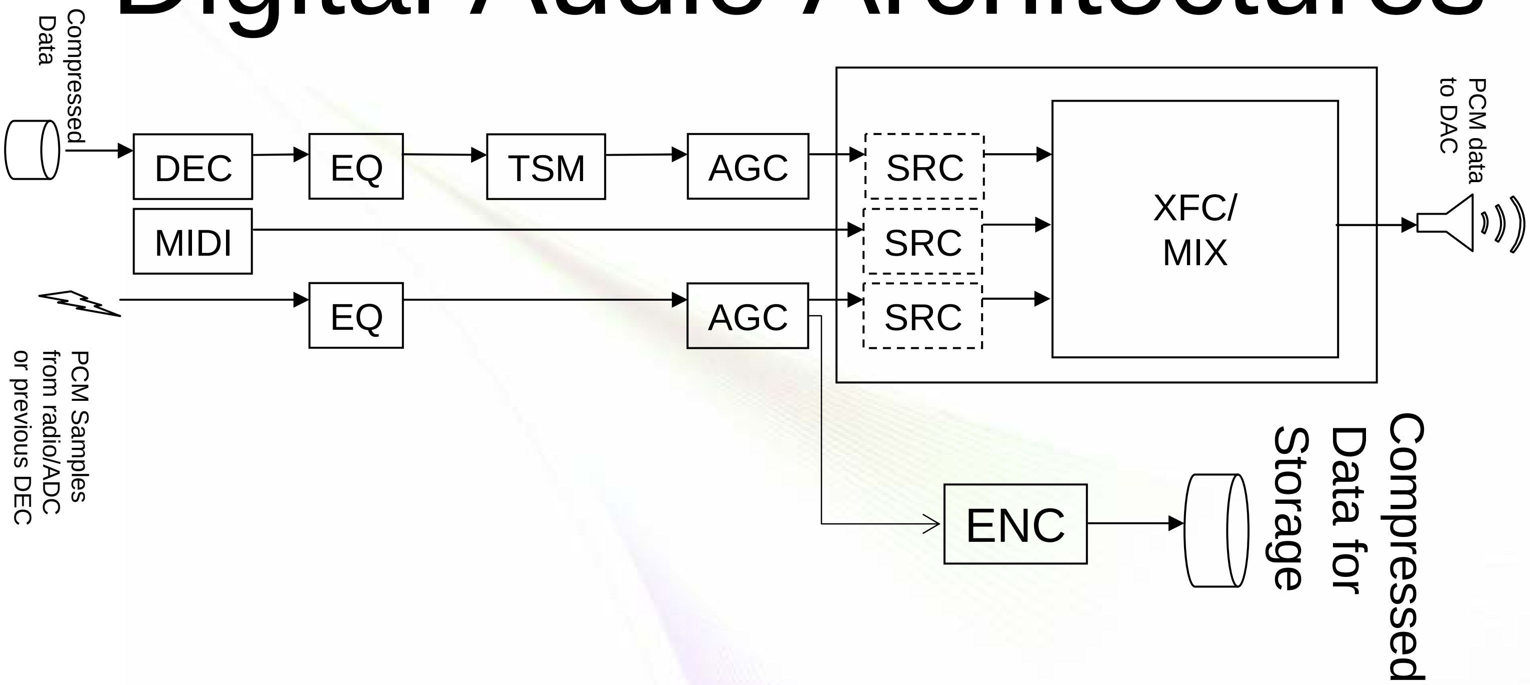
Decryptor: Decryption module like WMDRM10

Decoder: Audio decoder like MP3, AAC, WMA

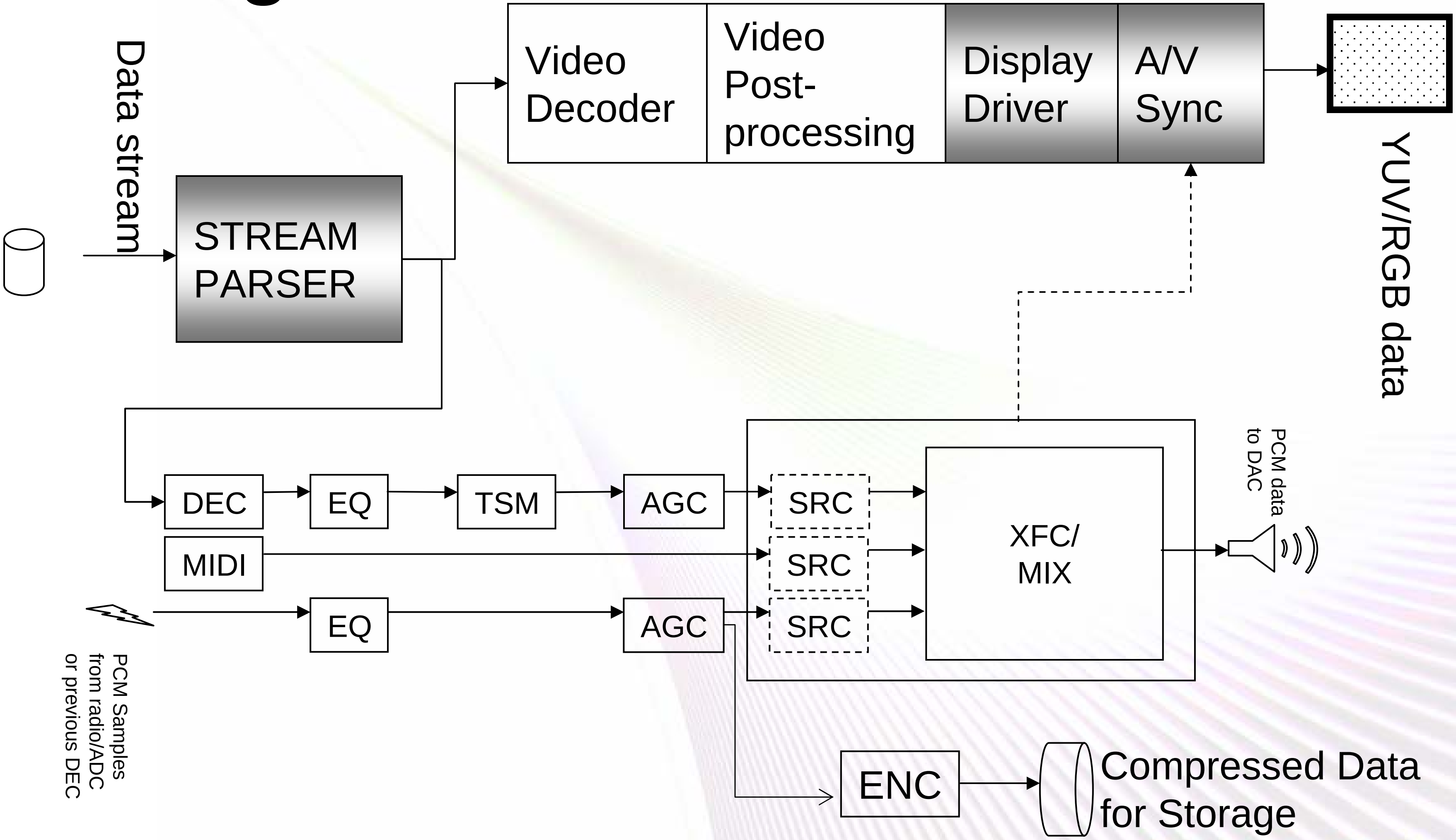
Effects: Algorithms for audio effects like Equalizer, TSM

Renderer: DAC Driver for PCM audio samples playback

Digital Audio Architectures

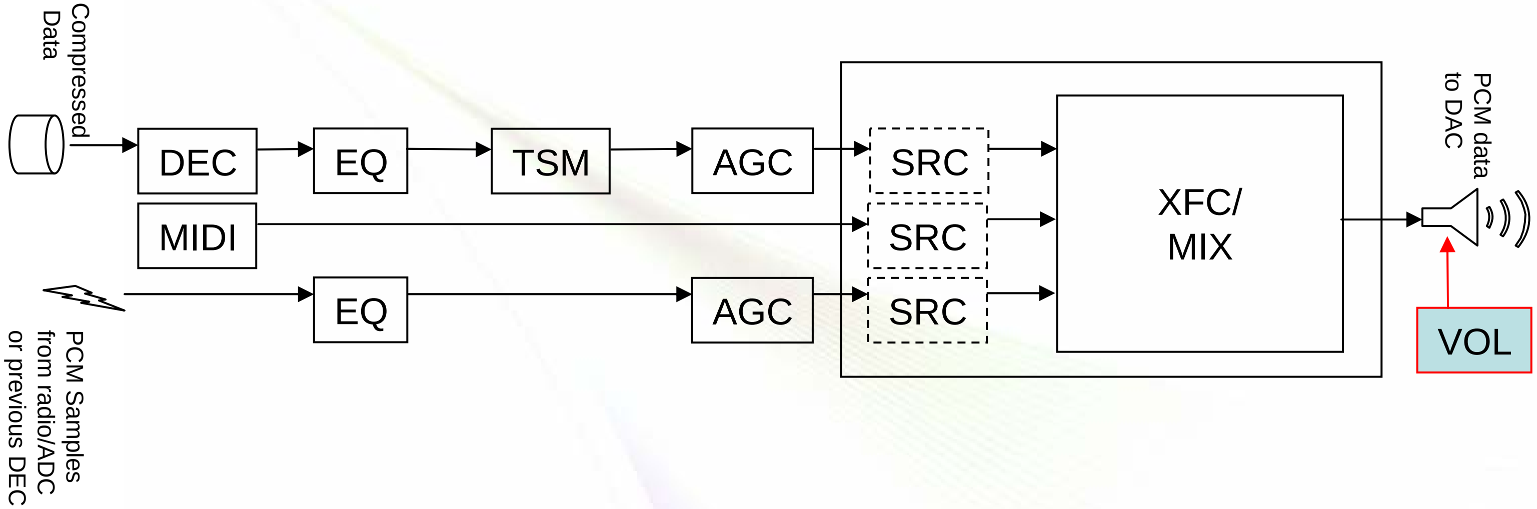


Digital Audio Architectures



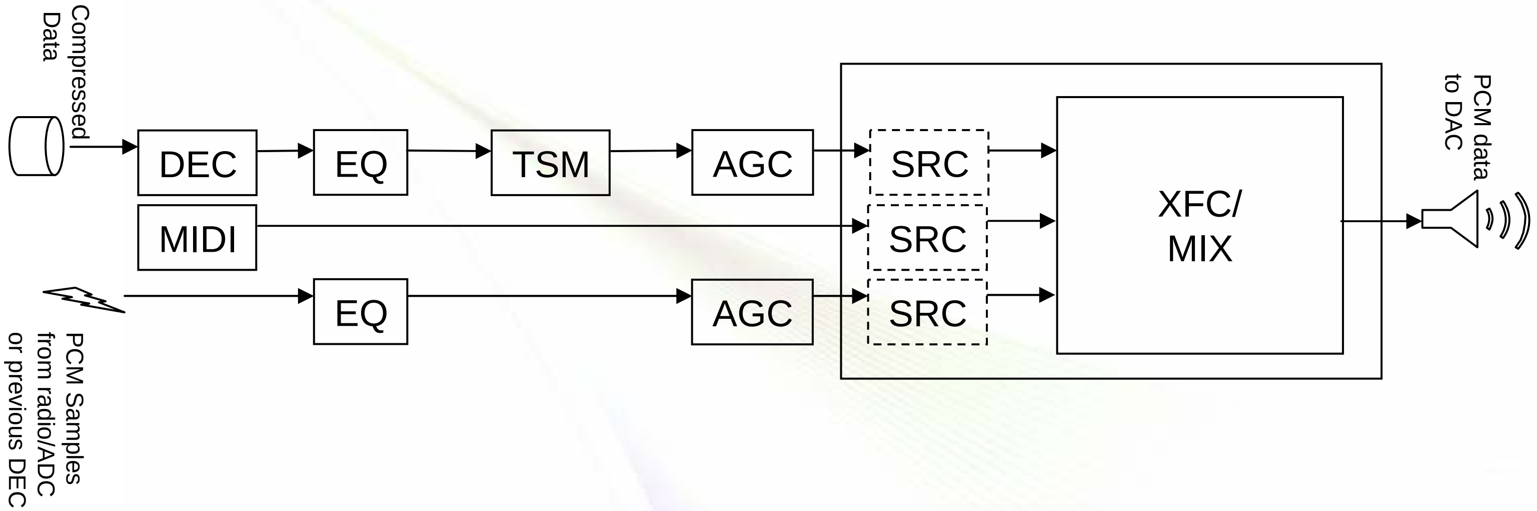
Minds in Motion

Digital Audio Design Challenges



- Saturation and dynamic range trade-offs
  - Limited bits per sample in fixed-point architecture
  - Gains from mixing could cause saturation
  - Gain reduction avoids saturation, but…
    - hurts dynamic range
    - perceived by user
- Synchronous changes to DAC gain
  - Needed to compensate for gain reductions in avoiding saturation

Digital Audio Design Challenges



- OEMs might choose different effects
  - Change number of channels/streams
  - Remove AGC, too many cycles
  - Perform EQ after XFC/MIX

Digital Audio Design Challenges

- Unexpected end of stream
  - Corrupted bit-stream or real-time error
  - User activity, such as stop or pause
- Frequent messages add MIPS and complexity
  - Avoid needing to send a message for every gain change
- No single idea of a good crossfade
  - Club-DJ might want slow/transparent transition
  - Radio-DJ might want long decay and quick ramp

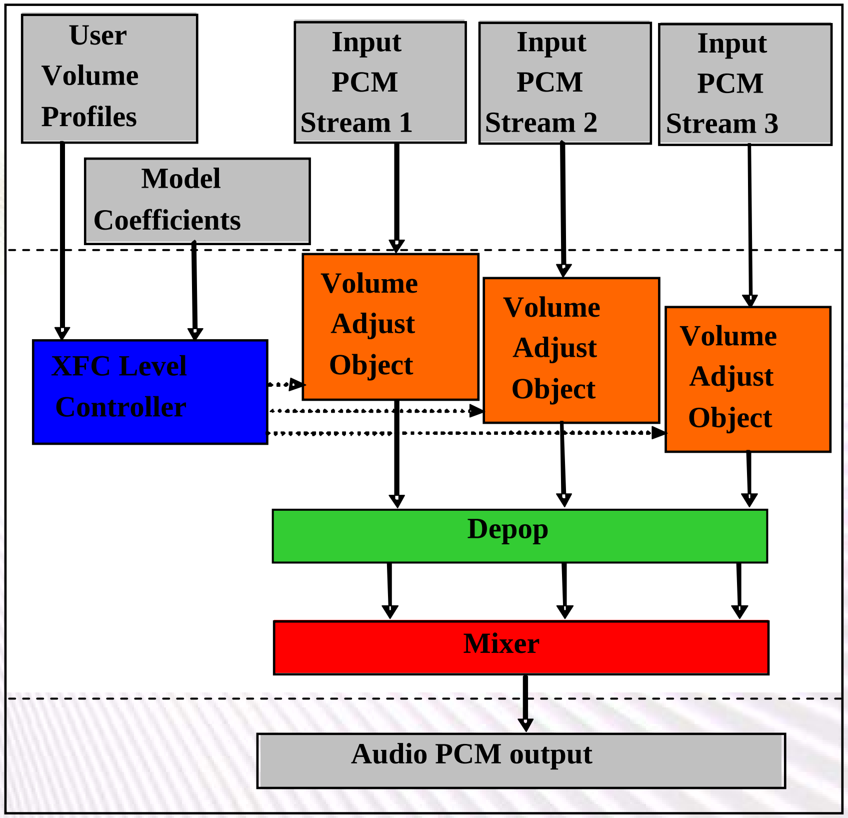
Crossfade Controller Features

- Volume level control for N streams
- User control of relationships between streams
- Automatic changes in volume targets based upon stream relationships
- Multi-segment volume profiles to emulate virtually any possible fade shape
- Depop logic removes “pops” from sudden stops in the audio
- Real-time events for fades, addition, and deletion of streams

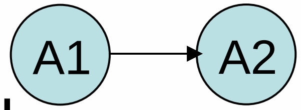
XFC Algorithm Architecture

- Crossfade module inputs from application:
  - User Volume Profiles
    - Set desired multi-segment volume fades for each stream
  - Model Coefficients
    - Defines stream relationships
  - Input PCM data for each stream
- Crossfade module contains:
  - XFC Level Controller
    - Uses volume profiles and model coefficients to determine how to alter Volume Adjust Objects
  - Volume Adjust Objects
    - Control/change volume for each of N streams (2*N objects for stereo processing)
  - Depop
    - Smooths out noise from sudden stops in audio
  - Mixer
    - Final summing of streams

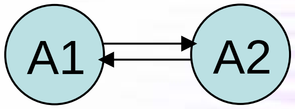
Figure 1: Structure of the XFC algorithm



Types of Stream Relationships



- Hierarchical
  - Volume of only one stream alters volume of other stream, without being affected by other stream (i.e. high vs. low priority)
  - Example: Priority(V_1)>Priority(V_2) $V_1' = V_1$ $V_2' = (1 - V_1)V_2$
  - Generic form: $V_2' = (\alpha_0 + \alpha_1 V_1)V_2$



- Equal
  - Volumes of two streams affect each other equally
  - Example: Priority(V_1)=Priority(V_2)

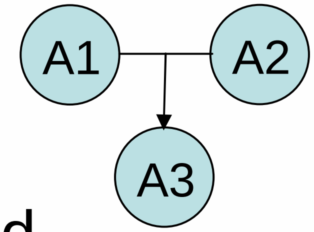
$$V_1' = \frac{1}{1 + V_2} V_1 \qquad V_2' = \frac{1}{1 + V_1} V_2$$

  - Generic form:

$$V_2' = \frac{1}{\beta_0 + \beta_1 V_1} V_2$$

Types of Stream Relationships (cont'd)

- Multiple influence



  - Volume of two or more streams equally alter volume of other stream, without being affected by other stream.
  - Example: Priority(V$_1$)=Priority(V$_2$)>Priority(V$_3$)

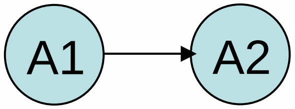
$$V_1' = \frac{1}{1+V_2} V_1 \qquad V_2' = \frac{1}{1+V_1} V_2 \qquad V_3' = (1 - V_1' - V_2')V_3$$

  - Generic Form:

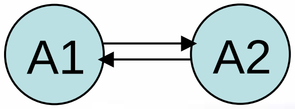
$$V_3' = (\gamma_0 + \gamma_{1_1} V_1' + \gamma_{1_2} V_2')V_3 \rightarrow V_3' = (\gamma_0 + \begin{bmatrix} \gamma_{1_1} & \gamma_{1_2} \end{bmatrix} \begin{bmatrix} V_1' \\ V_2' \end{bmatrix})V_3$$

Stream Relationships for XFC

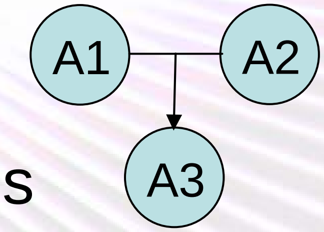
- Hierarchical
  - α_0 , α_1 used to generate terms
  - b_α bitmap matrix designates to which streams to apply these terms



- Equal
  - β_0 , β_1 used to generate terms
  - b_β bitmap matrix designates to which streams to apply these terms



- Multiple influence
  - γ_0 , γ_1 used to generate terms
  - b_γ bitmap matrix designates to which streams to apply these terms



Setting XFC Stream Relationships

Stream ID Number	Priority Level
1	1=Highest
2	1
3	2
4	3

$$g = \begin{Bmatrix} 1/(1+v_2) \\ 1/(1+v_1) \\ (1-v_1'-v_2') \\ (1-v_1'-v_2')(1-v_3) \end{Bmatrix}$$

$$g = \underbrace{\begin{Bmatrix} 1 \\ 1 \\ 1 \\ (1-v_3) \end{Bmatrix}}_{g_\alpha} \cdot \underbrace{\begin{Bmatrix} 1 \\ 1 \\ (1-v_1'-v_2') \\ (1-v_1'-v_2') \end{Bmatrix}}_{g_\gamma} \cdot \underbrace{\begin{Bmatrix} 1/(1+v_2) \\ 1/(1+v_1) \\ 1 \\ 1 \end{Bmatrix}}_{g_\beta}$$

Setting XFC Stream Relationships

Gain matrix	Terms	Bit Matrix
$g_\alpha = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ (1-v_3) \end{Bmatrix}$	$t_\alpha = \{1 - v_3\} = \underbrace{\{0 \quad 0 \quad -1 \quad 0\}}_{\alpha_1} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix} + \underbrace{\{1\}}_{\alpha_0}$	$b_\alpha = \{0 \quad 0 \quad 0 \quad 1\}$
$g_\beta = \begin{Bmatrix} 1/(1+v_2) \\ 1/(1+v_1) \\ 1 \\ 1 \end{Bmatrix}$	$t_\beta = \dfrac{1}{\begin{Bmatrix} 1+v_2 \\ 1+v_1 \end{Bmatrix}} = \dfrac{1}{\underbrace{\begin{Bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{Bmatrix}}_{\beta_1} \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{Bmatrix} + \underbrace{\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}}_{\beta_0}}$	$b_\beta = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{Bmatrix}$
$g_\gamma = \begin{Bmatrix} 1 \\ 1 \\ (1-v_1'-v_2') \\ (1-v_1'-v_2') \end{Bmatrix}$	$t_\gamma = \{1 - v_1' - v_2'\} = \underbrace{\{-1 \quad -1 \quad 0 \quad 0\}}_{\gamma_1} \begin{Bmatrix} v_1' \\ v_2' \\ v_3' \\ v_4' \end{Bmatrix} + \underbrace{\{1\}}_{\gamma_0}$	$b_\gamma = \{0 \quad 0 \quad 1 \quad 1\}$

Generating Stream Relationships from Priority Level

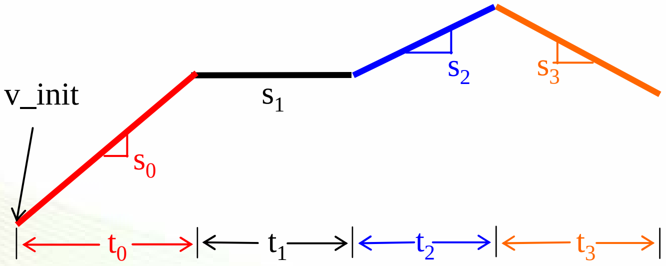
- Stream relationships can be explicitly set in model coefficients
- Or if strict priority-based structure is desired, model coefficients can be generated through XFC's XFC_coeff_gen() function

```
#define NUM_STREAMS 4
Int16 coeff_gen_p = {1, 1, 2, 3};
XFC_vol_adj_params model_coeff;

XFC_coeff_gen(NUM_STREAMS, coeff_gen_p, model_coeff);
```

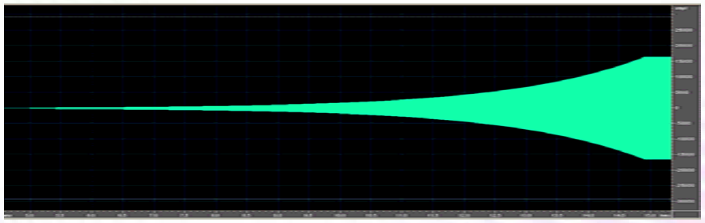
User Volume Profiles

- Multi-segment user volume profiles provide great flexibility in shapes of fades
  - Any shape can be emulated
  - Not just limited to linear or exponential decays
- User volume profiles consist of seven parts
  - `stream_id` = unique ID for each stream
  - `num_events` = number of segments in user volume profile
  - `del_t` = array that holds lengths of all segments (i.e. {t_0, t_1, t_2, t_3})
  - `s` = array that holds slopes of all segments (i.e. {s_0, s_1, s_2, s_3})
  - `v_init` = the initial value for start of the volume profile
  - `status` = the current state for the stream
  - `end_of_profile_status` = designates the `status` when the end of the user volume profile is reached.

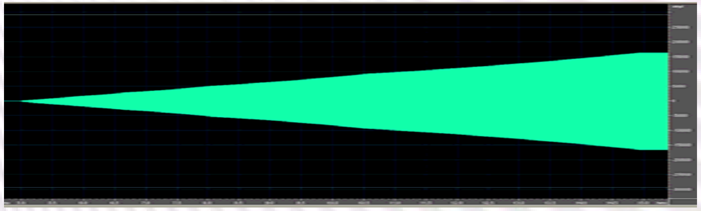
Example of User Volume Profile Shape



Single-Segment, Exponential Fade-In



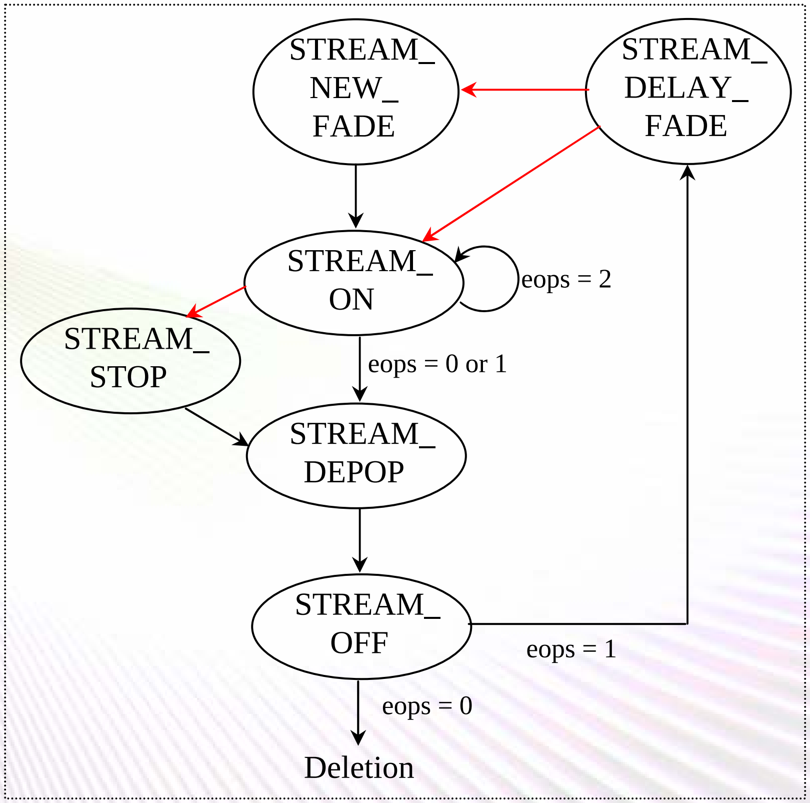
Multi-Segment, Linear Fade-In



XFC State Machine

- Exclusive state machine for each stream
- States
  - STREAM_DELAY_FADE – stream inactive and not mixed in, state preserved
  - STREAM_NEW_FADE – user-initiated state to designate start of a new user volume profile
  - STREAM_ON – stream active and mixed in
  - STREAM_STOP – user-initiated hard stop
  - STREAM_DEPOP – applying depop logic
  - STREAM_OFF – stream inactive and not mixed in, automatic transition to deletion or STREAM_DELAY_FADE

State Transition Diagram for XFC module

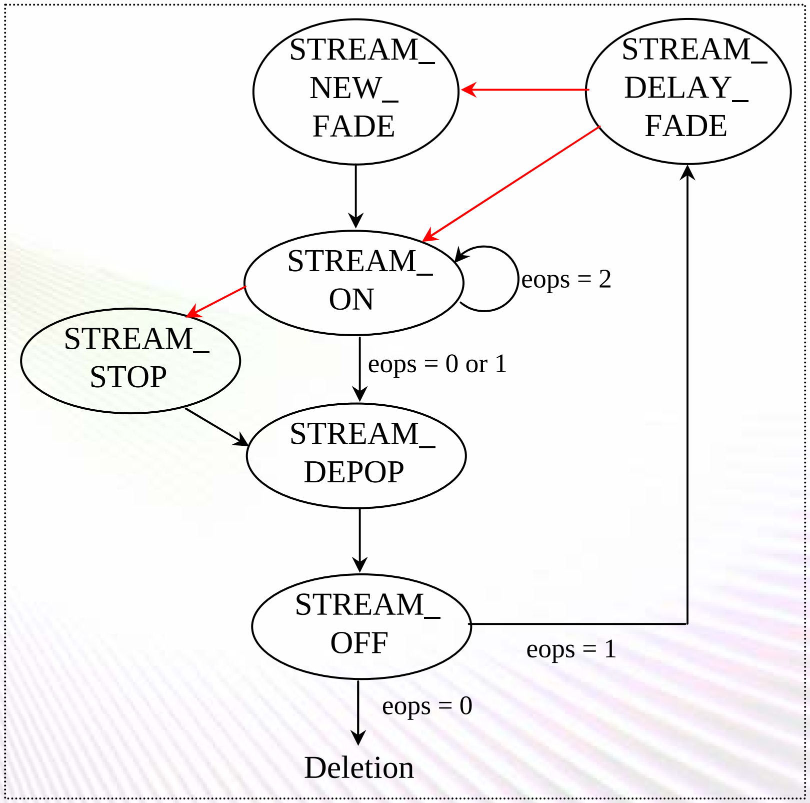


Red designates user-triggered state changes, while all other state changes occur automatically within the XFC algorithm.

End Of Profile Status Flag

- End of Profile Status Flag (eops) – means of automatically designating the status upon completion of user volume profile (i.e. no user intervention needed at end of fade)
- Three possible flag values
  - 0 = delete the stream upon completion of fade
  - 1 = stream is resident and silent upon completion of fade
  - 2 = stream is active and playing upon completion of fade

State Transition Diagram for XFC module

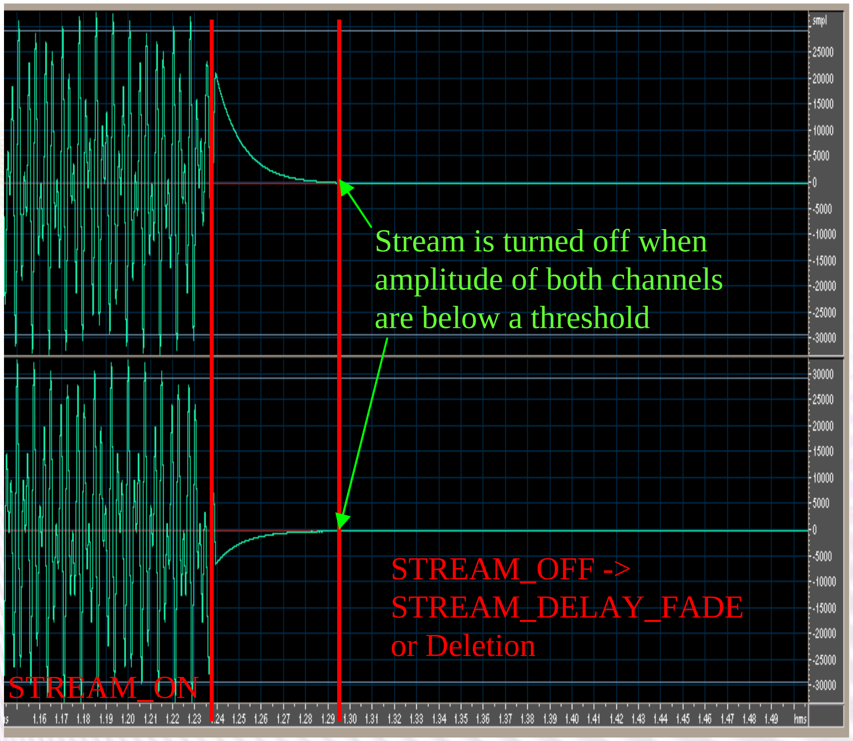


Red designates user-triggered state changes, while all other state changes occur automatically within the XFC algorithm.

Depop Processing

- Problem: “Pop” sound from discontinuity at stop of audio
  - Bitstream corruption or real-time error causes stop in middle of song without warning
  - User stops in middle of song without warning
  - End of song reached with no fade-out
  - End of song reached with partial fade-out
- Solution: Depop processing lessens discontinuity through exponential decay

Example of Depop Processing



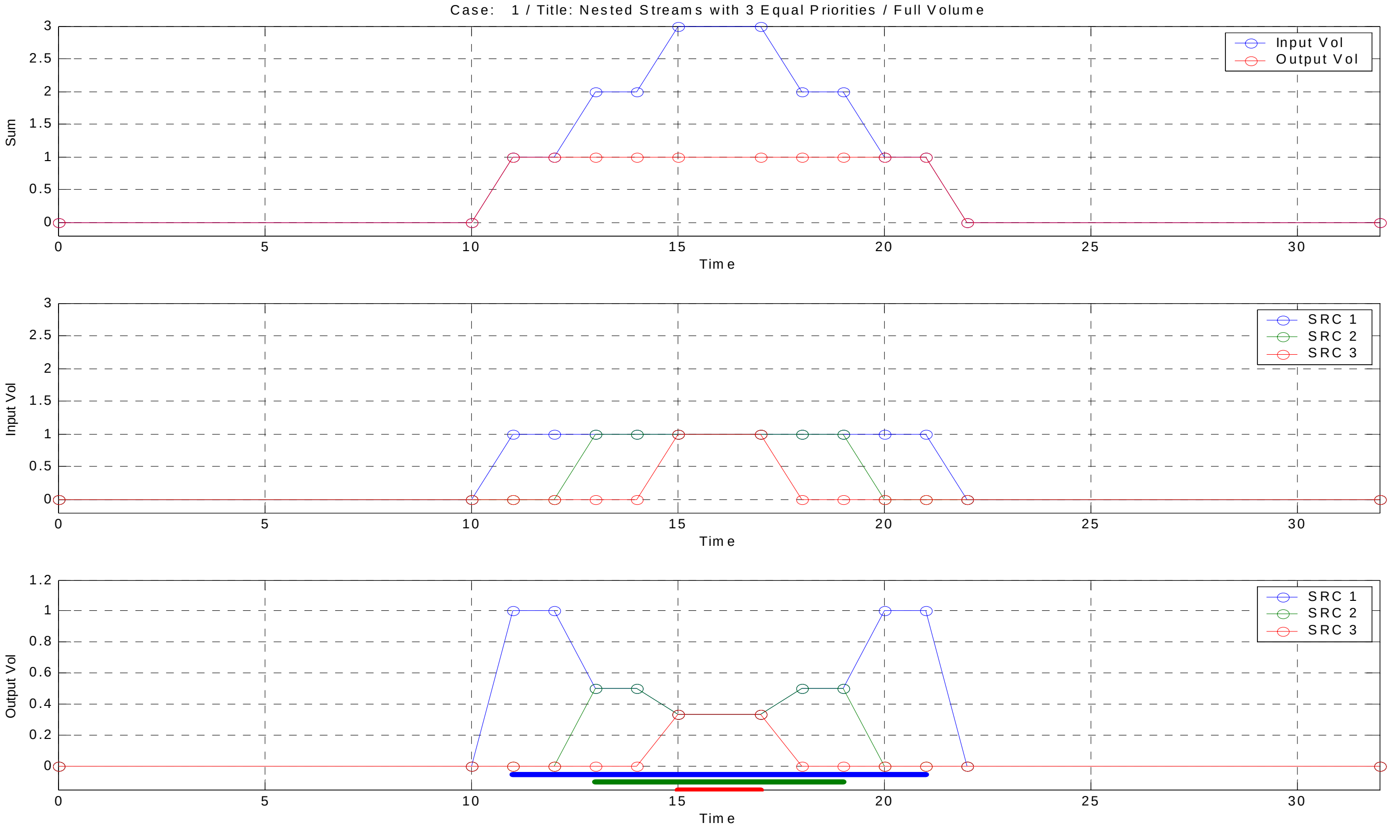
Stream is turned off when amplitude of both channels are below a threshold

STREAM_OFF -> STREAM_DELAY_FADE or Deletion

STREAM_ON

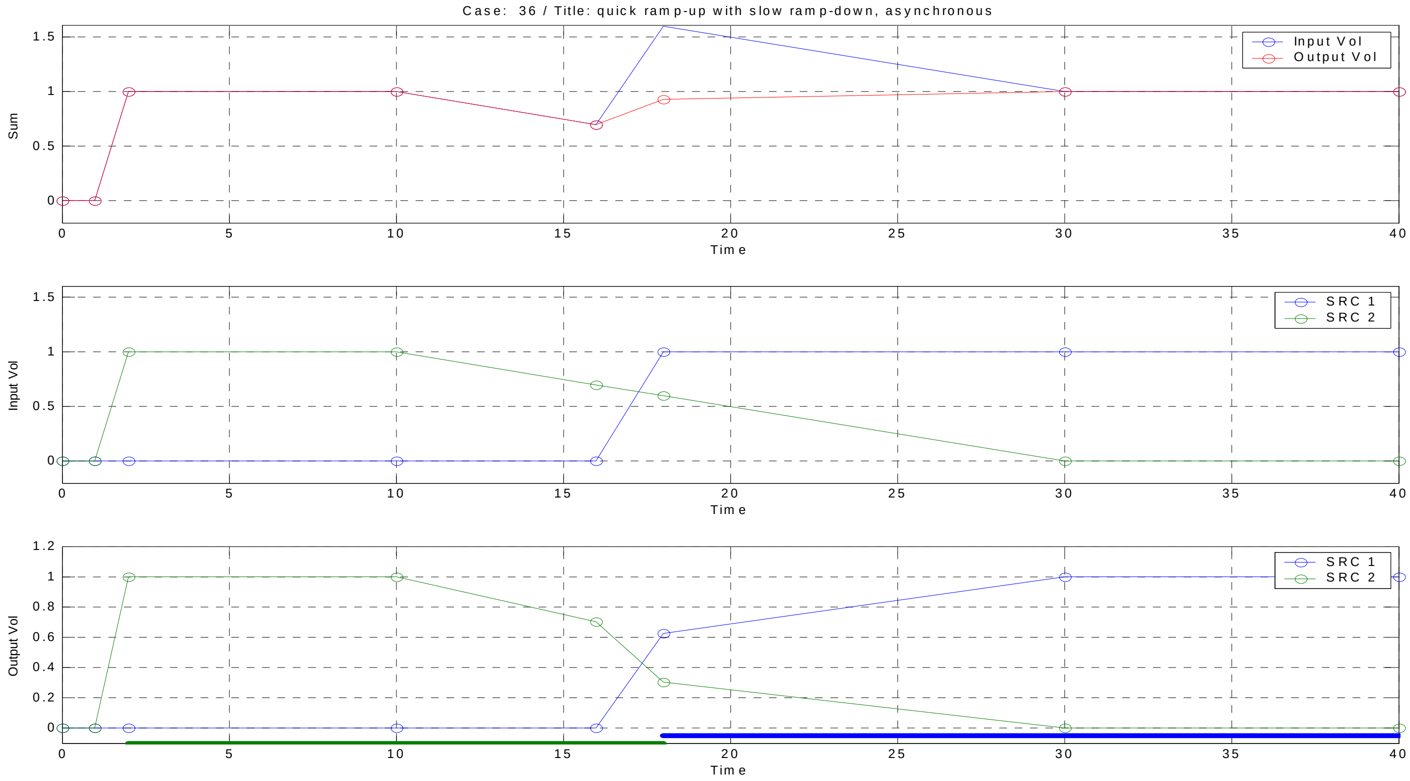
STREAM_DEPOP

Motivation for Event-Driven XFC: Multiple Streams



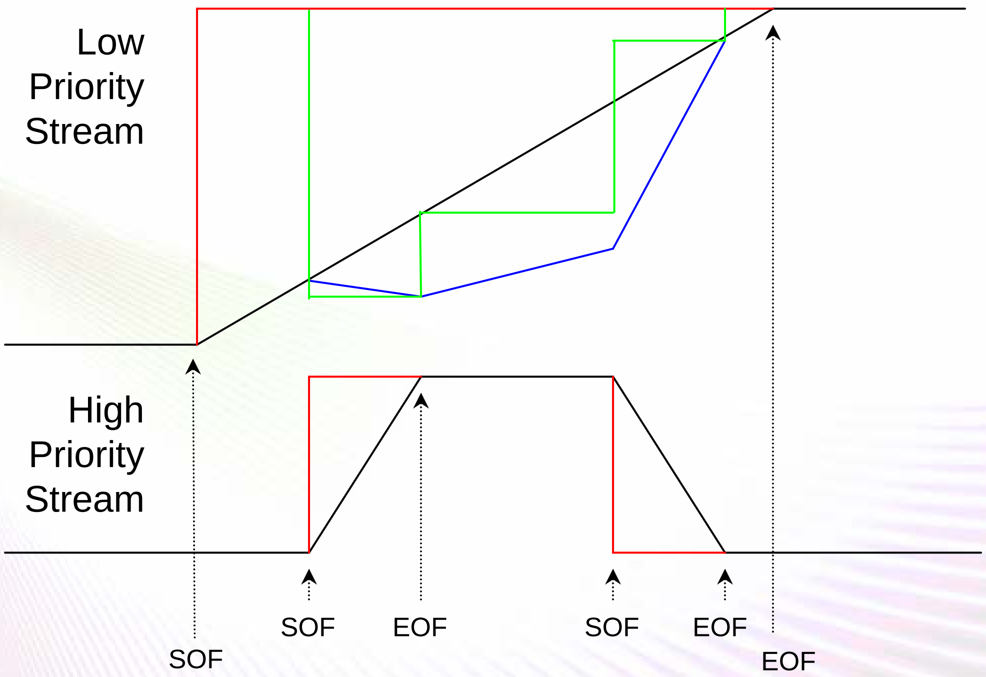
Volume Levels/Targets for each stream need to change to accommodate unanticipated streams

Motivation for Event-Driven XFC: Asynchronous Fades



Volume Targets and Events

- XFC is event-driven in order to handle interruptions by new streams
- At each real-time event, target volumes and delta times are recalculated based on stream relationships
- Events
  - Start of Fade (SOF) – beginning of user volume profile
  - End of Fade (EOF) – conclusion of user volume profile
  - End of Segment (EOS) – end of segment within user volume profile
- XFC controller keeps track of time to next event



Target Calculator Operating Example

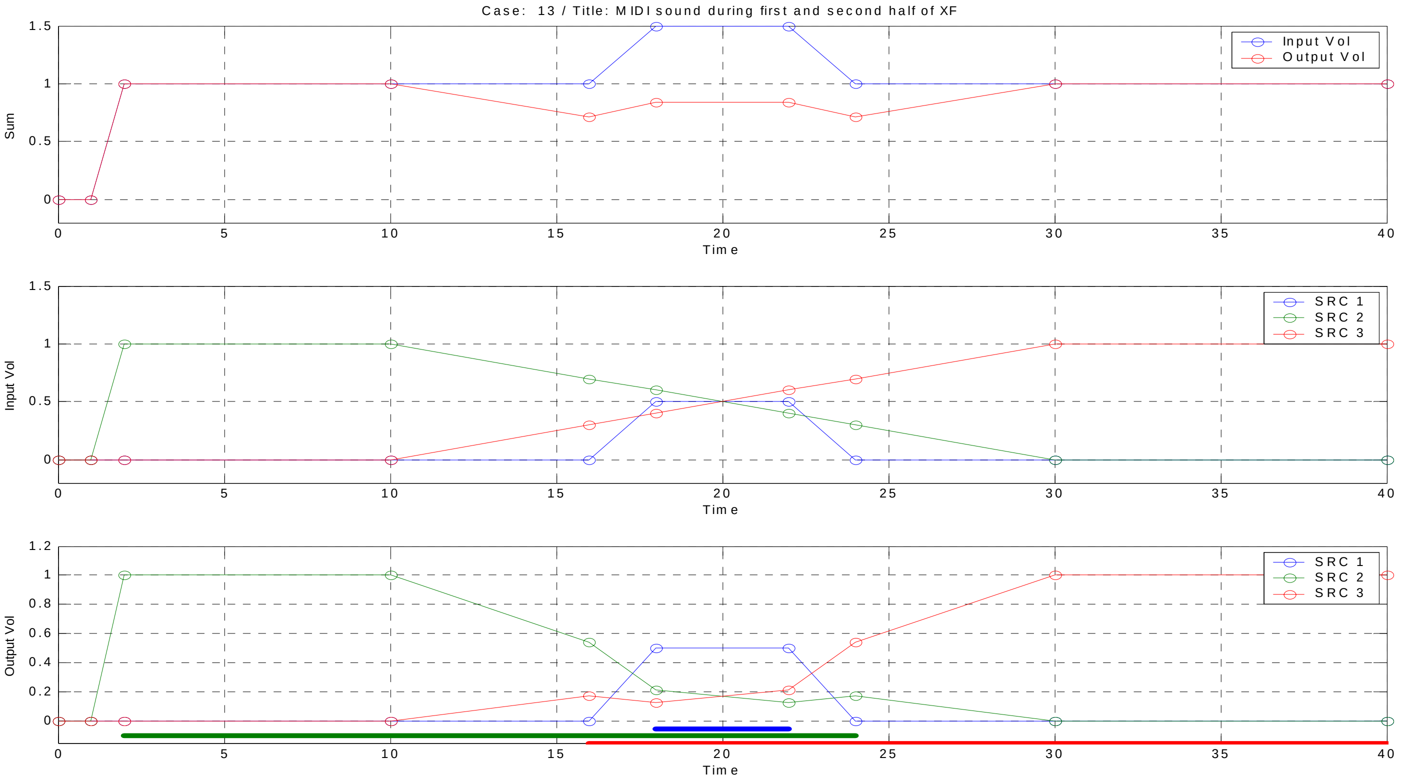
Original Volume Profile

Adjusted Volume Profile

Original Targets

Adjusted Targets

Common Scenario: Crossfade with UI sound



Implementation Steps by Client

1. Setup model coefficients for three streams
2. Create user volume profiles (fade shapes and volume targets) for each stream
3. Input model coefficients and user volume profiles into XFC
4. When desired, change stream status to STREAM_NEW_FADE to begin volume ramps

Common Scenario: XFC Stream Relationships

Stream ID Number	Priority Level
1	1=Highest
2	2
3	2

$$g = \begin{Bmatrix} 1 \\ (1-v_1)/(1+v_3) \\ (1-v_1)/(1+v_2) \end{Bmatrix}$$

$$g = \underbrace{\begin{Bmatrix} 1 \\ 1-v_1 \\ 1-v_1 \end{Bmatrix}}_{g_\alpha} \cdot \underbrace{\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}}_{g_\gamma} \cdot \underbrace{\begin{Bmatrix} 1 \\ 1/(1+v_3) \\ 1/(1+v_2) \end{Bmatrix}}_{g_\beta}$$

Or generate model coefficients automatically:

```
#define NUM_STREAMS 3
Int16 coeff_gen_p = {1, 2, 2};
XFC_vol_adj_params model_coeff;

XFC_coeff_gen(NUM_STREAMS, coeff_gen_p, model_coeff);
```

Common Scenario: XFC Stream Relationships

Gain matrix	Terms	Bit Matrix
$g_\alpha = \left\{\begin{matrix} 1 \\ (1-v_1) \\ (1-v_1) \end{matrix}\right\}$	$t_\alpha = \{1 - v_1\} = \underbrace{\{-1 \quad 0 \quad 0\}}_{\alpha_1} \left\{\begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix}\right\} + \underbrace{\{1\}}_{\alpha_0}$	$b_\alpha = \{0 \quad 1 \quad 1\}$
$g_\beta = \left\{\begin{matrix} 1 \\ 1/(1+v_3) \\ 1/(1+v_2) \end{matrix}\right\}$	$t_\beta = \dfrac{1}{\left\{\begin{matrix} 1+v_3 \\ 1+v_2 \end{matrix}\right\}} = \dfrac{1}{\underbrace{\left\{\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}\right\}}_{\beta_1} \left\{\begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix}\right\} + \underbrace{\left\{\begin{matrix} 1 \\ 1 \end{matrix}\right\}}_{\beta_0}}$	$b_\beta = \left\{\begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}\right\}$
$g_\gamma = \left\{\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}\right\}$	$t_\gamma = \{1\} = \underbrace{\{0 \quad 0 \quad 0\}}_{\gamma_1} \left\{\begin{matrix} v_1' \\ v_2' \\ v_3' \end{matrix}\right\} + \underbrace{\{1\}}_{\gamma_0}$	$b_\gamma = \{0 \quad 0 \quad 0\}$

Common Scenario: Setting Up User Volume Profiles

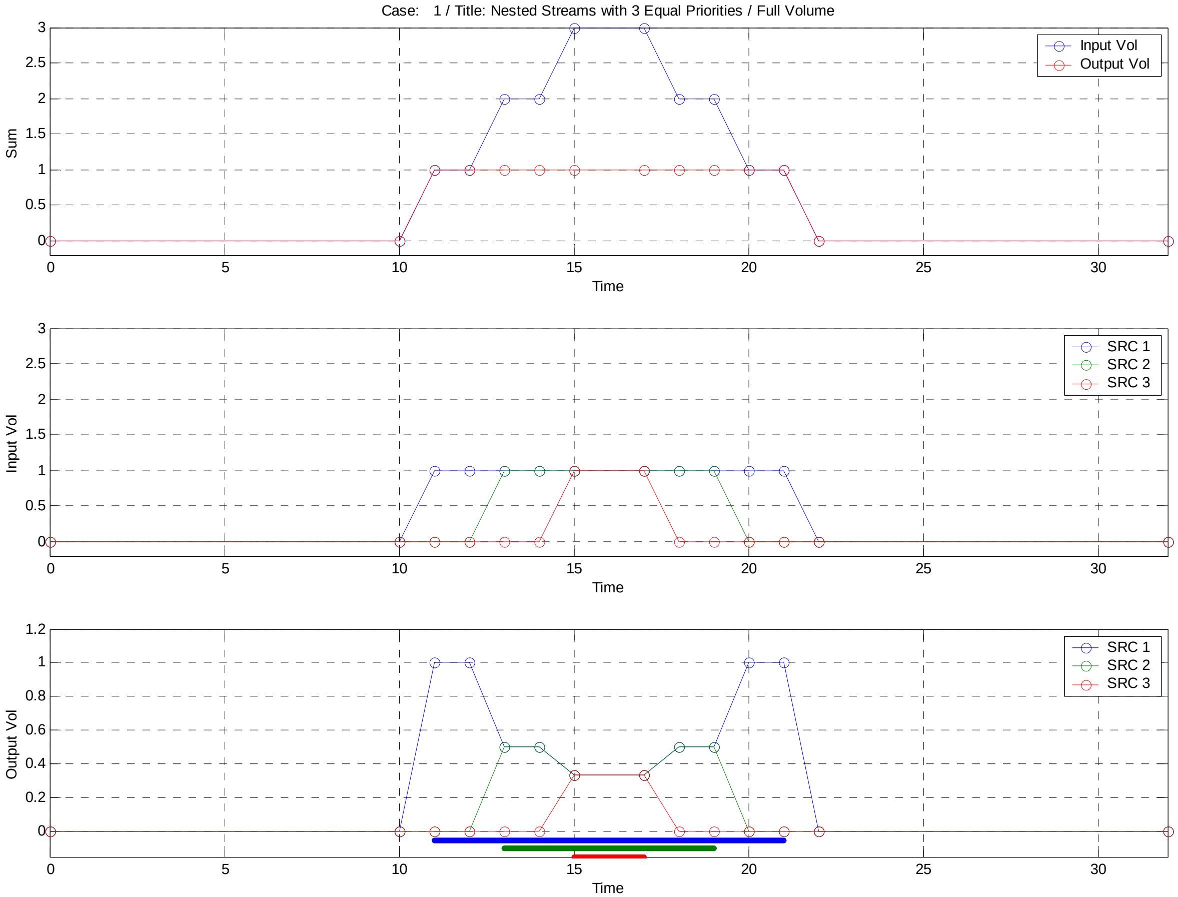
```
XFC_user_volume_profile slow_linear_fadeout =
{
        1,                          // stream id
        MULTISEGMENT_NUM1,          // num_events
        slow_linear_fadeout_del_t   // del_t
        slow_linear_fadeout_s,      // s
        - 32768,                    // v_init
        STREAM_NEW_FADE,            // stream state
        1                           // end of profile status
};

XFC_user_volume_profile fast_linear_fadein =
{
        3,                          // stream id
        MULTISEGMENT_NUM2,          // num_events
        fast_linear_fadein_del_t    // del_t
        fast_linear_fadein_s,       // s
        - 10240,                    // v_init (Q8.8) (-40dB)
        STREAM_DELAY_FADE,          // stream state
        2                           // end of profile status
};
```

```
XFC_fade(XFCHandle, xfcStatus,
fast_linear_fadein);
XFC_fade(XFCHandle, xfcStatus,
slow_linear_fadeout);
XFC_fade(XFCHandle, xfcStatus,
slow_linear_fadein);

…..

if (time==later)
{
    fast_linear_fadein.status =
        STREAM_NEW_FADE;

    XFC_fade(XFCHandle,xfcStatus,
        fast_linear_fadein);
}
```

Results

- Flexible level/crossfade controller
  - User control of stream relationships
  - Multi-segment volume fades
  - Event-driven module
- Implemented on C55x DSP
- Performance
  - Memory (including Mixer and Volume objects)
    - Program = 9.2kB
    - Data = 0.4kW
    - Instance ~ 484 - 700 words
  - MIPS
    - Test Scenario = 5-second crossfade of two streams
    - Average = 4.0 MIPS
    - Worst Case = 8.1 MIPS

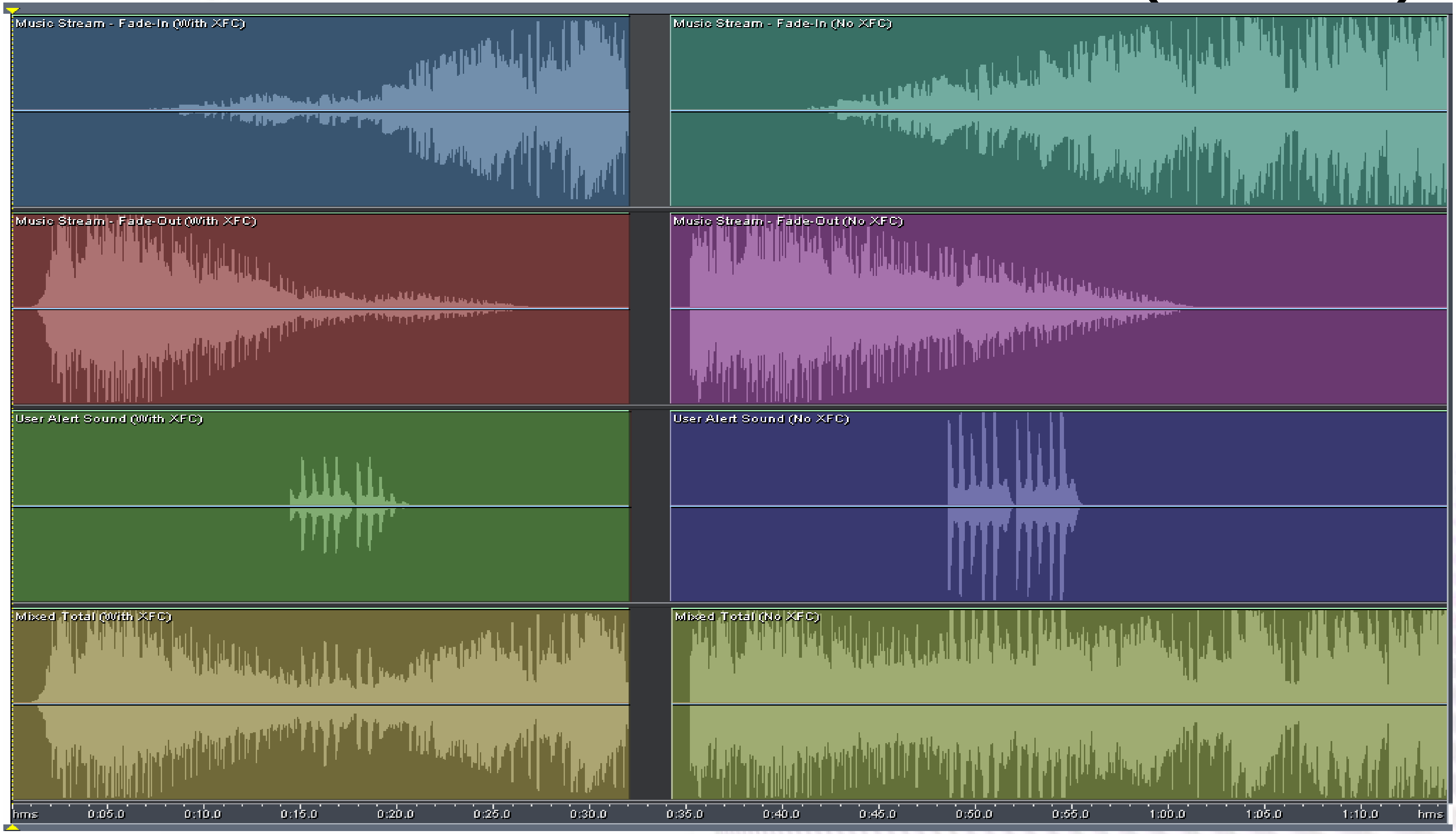
Demo – Equal Priority Sinusoids



Sum (No XFC)

Sum (With XFC)

Demo- XF with Alert Sound (Music)


Music Stream - Fade-In (With XFC)
Music Stream - Fade-In (No XFC)
Music Stream - Fade-Out (With XFC)
Music Stream - Fade-Out (No XFC)
User Alert Sound (With XFC)
User Alert Sound (No XFC)
Mixed Total (With XFC)
Mixed Total (No XFC)
hms
0:05.0
0:10.0
0:15.0
0:20.0
0:25.0
0:30.0
0:35.0
0:40.0
0:45.0
0:50.0
0:55.0
1:00.0
1:05.0
1:10.0
hms

With XFC

No XFC

Conclusions

- Discussed today
  - Need to handle multiple streams and crossfades
  - Audio System Architectures and Challenges
  - Crossfade/Level Controller Highlights
    - Stream Relationships
    - Construction of Volume Fades
    - Real-time event functionality
  - Implementation for Common Scenario
  - Results
  - Demonstrations
- Possible Future Work
  - Currently based on maximum volume of stream, could use current volume level of stream

TI Developer Conference

March 7-9, 2007 • Dallas, TX

Solving Audio Subsystem Challenges Using a Crossfade/Level Controller Algorithm in Portable Media Applications

Danny Jochelson
dsjochel@ti.com
Jason Kridner
jdk@ti.com

Minds in Motion