Real-time Audio Processing Capabilities of Microcontrollers, Application Processors, and DSPs

Paul Beckmann

DSP Concepts, LLC

Overview

- Motivation
- Comparing DSP and Microcontroller architectures
- FIR and IIR filter benchmarks
- Real-world examples
  - 2.1 channel speaker crossover
  - Automotive audio system
- Conclusion

Introduction and Motivation

- Media rich applications and products are proliferating
- Systems currently consist of multiple processors
  - Micros / Application processors – control and UI
  - Graphics processors – video and graphics
  - DSPs – audio processing
- In some cases they are integrated into a single SOC (“System on a Chip”)

- Can the audio processing tasks traditionally handled by a dedicated DSP be migrated to a microcontroller or an application processor?

The Architecture of a DSP

- Multiple memory buses
- Single cycle multiply - accumulate
- Zero-overhead loops
- Load and stores in parallel with computation
- Accumulator with guard bits
- Fractional and saturating math
- SIMD instructions for parallel computation
- Barrel shifter
- Floating-point hardware
- Circular and bit-reversed addressing

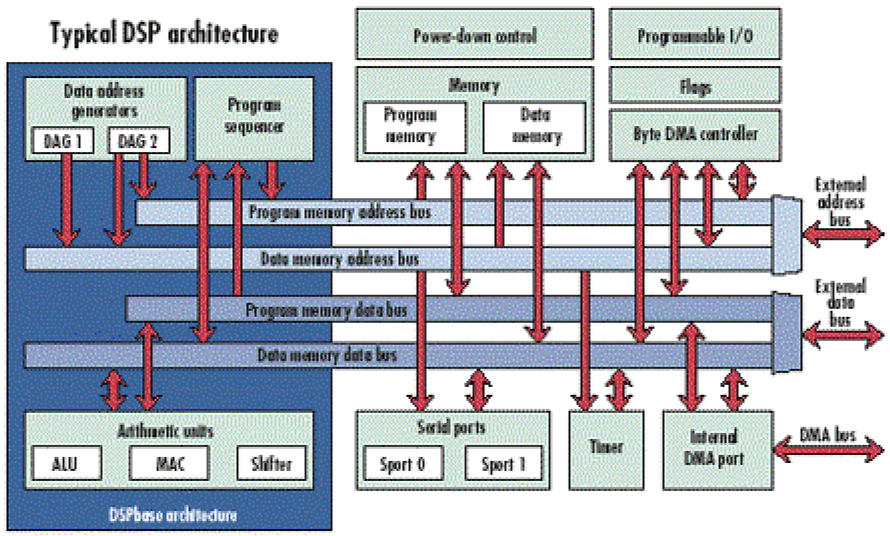


Figure taken from machinedesign.com

The Architecture of a Microcontroller

- Single memory bus
- MAC takes 4 to 7 cycles
- Loops overhead of 3 cycles
- Load/stores or computation
- No guard bits
- Integer math with overflow
- No SIMD
- No barrel shifter
- Floating-point hardware
- No circular and bit-reversed addressing

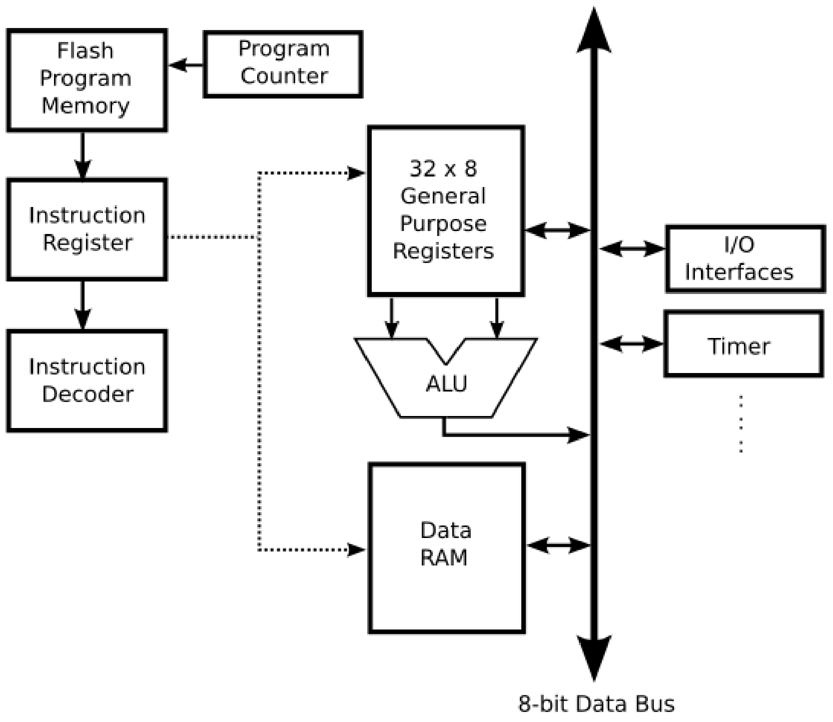


Figure taken from machinedesign.com

Other Noteworthy Differences

DSPs

- Large register file
- Serial ports
- Sample rate converters
- Flexible DMA controllers
- Require ASM programming to achieve maximum performance
- Lower power consumption (milliwatt per MIP)

Microcontrollers

- Small register file
- Cached architecture
- Low power sleep modes
- Low cost
- Large number of peripherals
- Many variants
- Integrated flash memory
- Good driver support
  - USB/Ethernet/CAN/Flash/etc
- Operating systems
- Many features fully programmable from C
- Low interrupt latency

Digital Signal Controllers

Digital Signal Controller = Microcontroller + DSP features

+ Multiple memory buses
+ Single cycle multiply - accumulate
– Zero-overhead loops
– Load and stores in parallel with computation
– Accumulator with guard bits
+ Fractional and saturating math
+ SIMD instructions for parallel computation
– Barrel shifter
– Circular and bit-reversed addressing

An example is the ARM Cortex-M4

Up to 180 MHz
Floating-point
USB
<$3 in volume

Other DSC families available from TI, Freescale, and Microchip

Application Processors

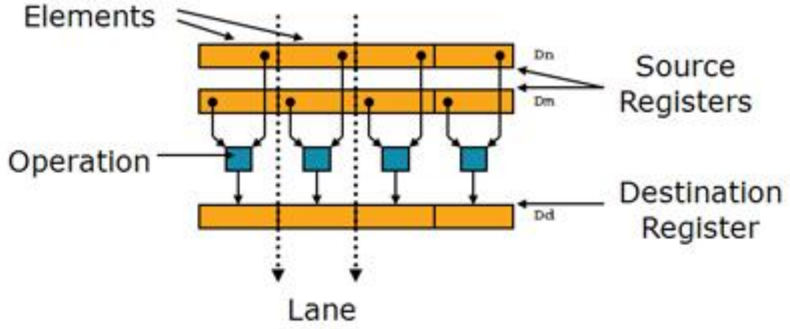
- High-end microcontrollers
- Clock speeds up to 2 GHz
- High level of integration
  - Multiple processor cores
  - Graphics coprocessor
  - Networking
  - USB
  - Security features

- Examples
  - ARM Cortex-A processor family
  - Intel Atom
- Used in
  - Smart phones
  - iPad/tablets
  - Set top boxes
  - Automotive "head units"

ARM Cortex-A NEON Technology

- General purpose SIMD engine targeted at audio and video processing
- Large register file viewed as
  - 32 x 64-bit registers
  - 16 x 128-bit registers
- 2- or 4-way floating-point SIMD
- Programmable using C intrinsics or ASM

NEON™



Collision Course

- DSPs are adding peripherals and increasing software support
  - Analog Devices Blackfin
  - TI C5000
- Microcontrollers and application processors are adding DSP instructions
  - ARM Cortex-M4 = M3 + DSP instructions
  - ARM Cortex-A8/A9 have NEON
- Which device will win out?
- Is it easier to retrofit a DSP or a microcontroller?

Who Will Win?

For DSPs to win they need:

- Lower power sleep modes
- Lower cost
- Larger number of peripherals
- More variants
- Integrated flash memory
- Good driver support
  - USB/Ethernet/CAN/Flash/etc
- Operating systems

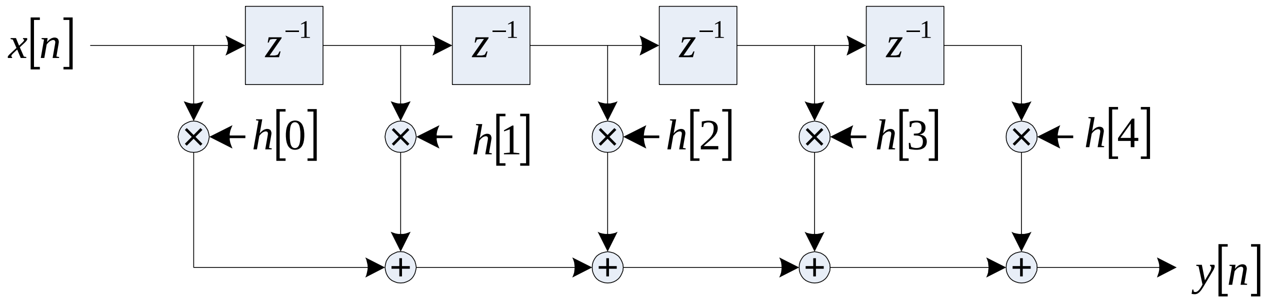
For Micros to win they need:

- Better power consumption
- Audio specific peripherals
  - Serial ports
  - Sample rate converters
  - Flexible DMA controllers
- High performance arithmetic

FIR Filter

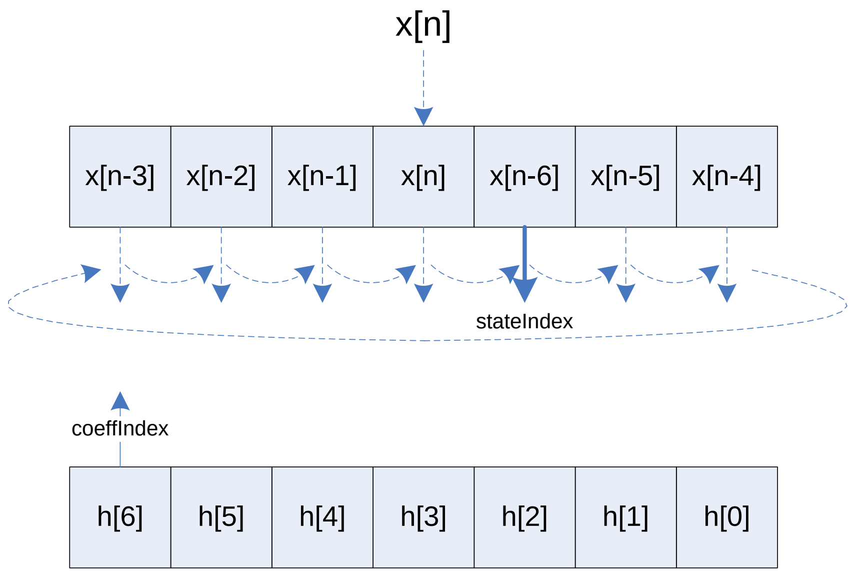
- Commonly used in
  - Audio processing
  - Video processing
  - Data smoothing
  - Communications
  - Control
- Benchmark DSP Algorithm
  - MACs
  - High memory bandwidth
  - Looping

$$y[n] = \sum_{k=0}^{N-1} x[n-k]h[k]$$



Circular Addressing

- Using a FIFO on a sample-by-sample basis is very inefficient
- Avoid data movement and use a circular buffer instead



State variables use a circular buffer

Coefficients use linear addressing

FIR Implementation – Simple C

```
for(sample=0;sample<blockSize;sample++)
{
  // Copy the new sample in
  state[stateIndex++] = inPtr[sample]
  if (stateIndex >= N)
    stateIndex = 0;

  sum = 0.0f;
  for(i=0;i<N;i++)
    {
      sum += state[stateIndex++] * coeffs[N-i];
      if (stateIndex >= N)
        stateIndex = 0;
    }
  outPtr[sample] = sum;
}
```

Code operates on a block of data

Inner loop is over N filter coefficients

FIR Implementation – DSP ASM

```
  lcntr = r2, do VEC_FIR_TapLoop until lce;
VEC_FIR_TapLoop:
  f12=f0*f4, f8=f8+f12, f4=dm(i1,m4), f0=pm(i12,m12);
```

- Executes in a single cycle!
  - Two data fetches
  - Multiplication
  - Addition
  - Circular addressing
  - Pointer updates
  - Looping

FIR Implementation – Cortex-M4

- Missing features
  - Multiple memory buses
  - Computation in parallel with memory accesses
  - Zero overhead loop
  - Circular memory addressing
- Work arounds
  - Cache state variables and coefficients and compute 4 outputs in parallel
  - Manually unroll the inner loop by a factor of 4
  - Use a FIFO but shift in data one block at a time
  - (Similar techniques apply to the Cortex-A8)

FIR Filter Benchmarks

	Measured Clock Cycles		
	DSP	**Cortex-M4**	**Cortex-A8**
Standard C	10386	46996	111721
Tuned C		17704	10330
Assembly	2974	13719	4238

64-point filter
64-sample block size

DSP and Cortex-A8 rely on SIMD

Cortex-M4 FIR library available from ARM (CMSIS DSP Library)

Cortex-A8 FIR library will be available from DSP Concepts

Standard C – Start with the textbook implementation of an algorithm and allow the C compiler to optimize as best as it can.

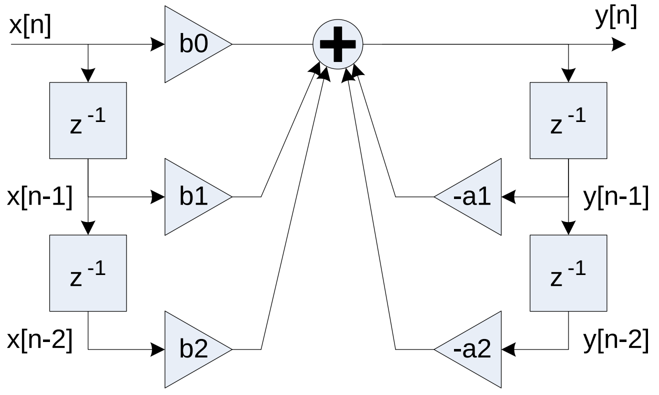
Tuned C – Hand optimize the code as best as possible while remaining in C. This involves loop unrolling, caching of variables, and using intrinsic functions.

Assembly – Get the absolute best performance possible using assembly coding.

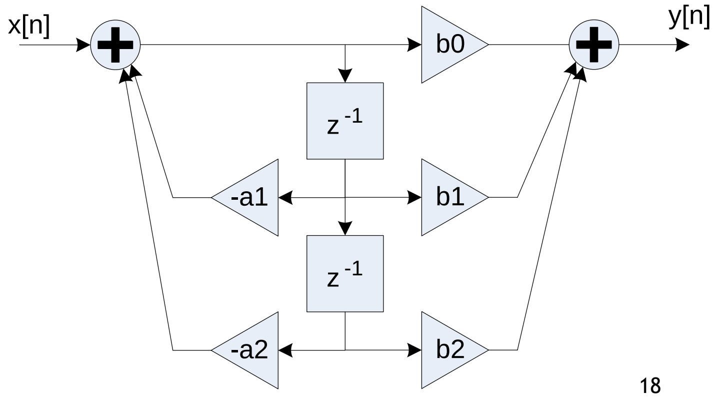
Biquad Filter

- Commonly used in audio:
  - Tone controls
  - Graphic EQ
  - Loudness compensation
  - Crossover filters
  - Etc.
- Different structures have advantages
  - Direct Form 1 – better fixed-point behavior
  - Direct Form 2 – less memory

Direct Form 1



Direct Form 2



Biquad Implementation – Simple C

```
// b0, b1, b2, a1, and a2 are the filter coefficients.
// a1 and a2 are negated.
// wNm1 and wNm2 represent the two state variables.

for (sample = 0; sample < blockSize; sample++)
  {
    wN = a1*wNm1 + a2*wNm2 + inPtr[sample]
    outPtr[sample] =  b0*wN + b1*wNm1 + b2*wNm2;
    wNm2=wNm1;
    wNm1=wN;
  }

// Persist state variables for next call
state[0] = wNm1;
state[1] = wNm2;
```

Code operates on a block of data

Inner loop has 5 multiplications

FIR Implementation – DSP ASM

```
lcntr=r1, do _sampleLoopEnd until lce;

// r15 = a2 * w[n-2].  r8 = src[i]
f15=f0*f11,r8=dm(i4,m4);

// r8 = a1 * w[n-1].  r15 = a2 * w[n-2] + src[i].
// dst[i-1] = result
f8=f3*f5,f15=f8+f15,pm(i12,m12)=r12;

// r10 = b2 * w[n-2].
// r2 = a1 * w[n-1] + a2 * w[n-2] + src[i] (= w[n])
// w[n-2] = w[n-1]
f10=f0*f6,f2=f8+f15,r0=r3;

// r8 = b0 * w[n]. r15 = b2 * w[n-2] + b1 * w[n-1]
// w[n-1] = w[n]
f8=f2*f7,f15=f10+f14,r3=r2;

_sampleLoopEnd:
//r14 = b1 * w[n-1], (new value for next loop iteration)
f14=f3*f4,f12=f8+f15;
```

- Ideal! Inner loop requires 5 instructions
- With SIMD can compute two filters in parallel

Biquad Implementation – Cortex-M4

- Missing features
  - Zero overhead loop
- Work arounds
  - Manually unroll the inner loop by a factor of 4
  - (Similar techniques apply to the Cortex-A8)

Biquad Filter Benchmarks

	Measured Clock Cycles		
	DSP	**Cortex-M4**	**Cortex-A8**
Standard C	2902	5503	18060
Tuned C		4812	4896
Assembly	1440	3840	2012

64 sample block size
Cascade of 4 filters

4 x 5 x 64 = 1280 MACs

No SIMD used in benchmarks

Standard C – Start with the textbook implementation of an algorithm and allow the C compiler to optimize as best as it can.

Tuned C – Hand optimize the code as best as possible while remaining in C. This involves loop unrolling, caching of variables, and using intrinsic functions.

Assembly – Get the absolute best performance possible using assembly coding.

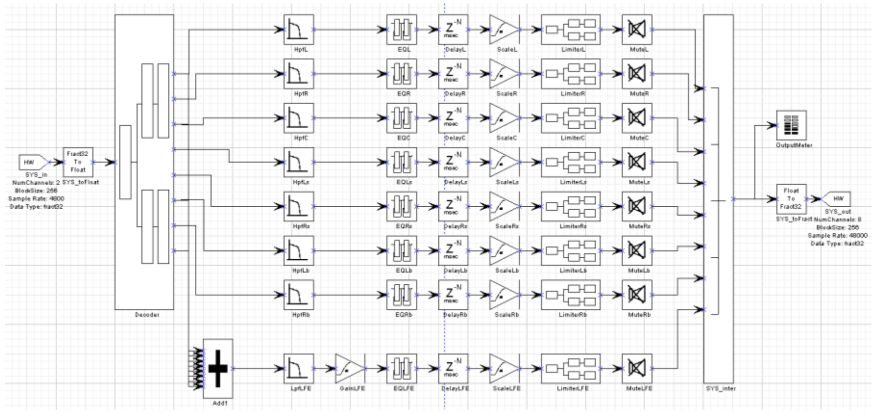
Real World Examples

- Compared two different systems
  - 2.1 channel loudspeaker processing
  - 13 channel automotive system
- Processors compared
  - Cortex-M4F. NXP LPC 43xx. 180 MHz
  - Cortex-A9. TI OMAP 4430. 1 GHz
  - 32-bit floating-point DSP. 400 MHz

Benchmarking With Audio Weaver

Complete SW solution for audio products

- Large library of optimized audio modules
- Supports SHARC, Blackfin, Cortex-M4, and Cortex-A8/9
- Built upon MATLAB
- Graphical drag-and-drop editor
- Real-time tuning
- Highly optimized for MIPs and memory usage

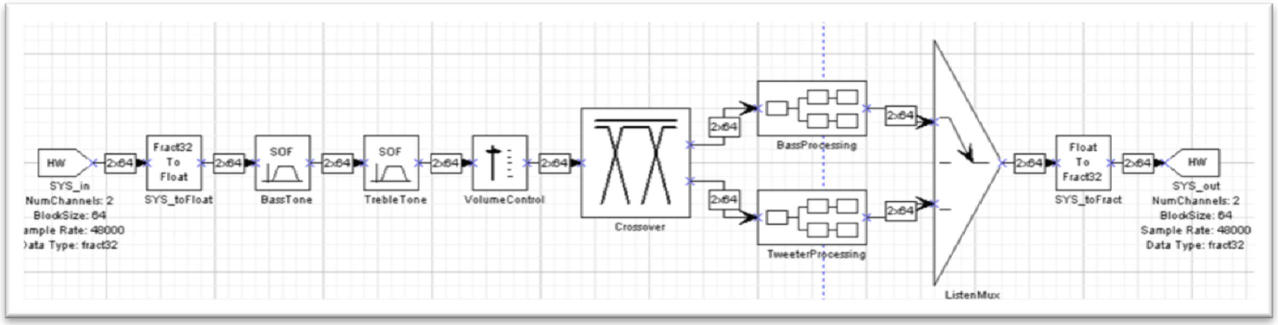


2.1 Channel Loudspeaker Processing

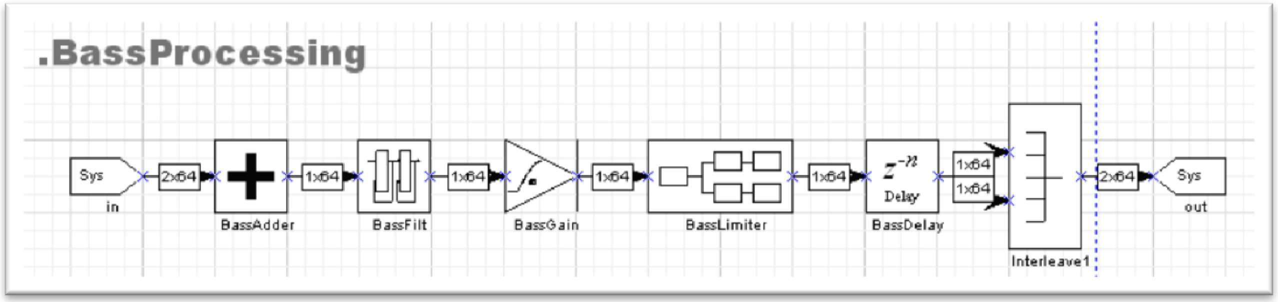
- Stereo multimedia loudspeakers
  - USB or analog inputs
  - 2.0 or 2.1 outputs
- Multimedia / gaming headphones
  - USB or analog input
  - Boom mic
  - Stereo in / stereo out
- iPod docking stations
  - USB or analog input
  - 2.0 or 2.1 or more outputs

Benchmarking results apply to all of these product categories

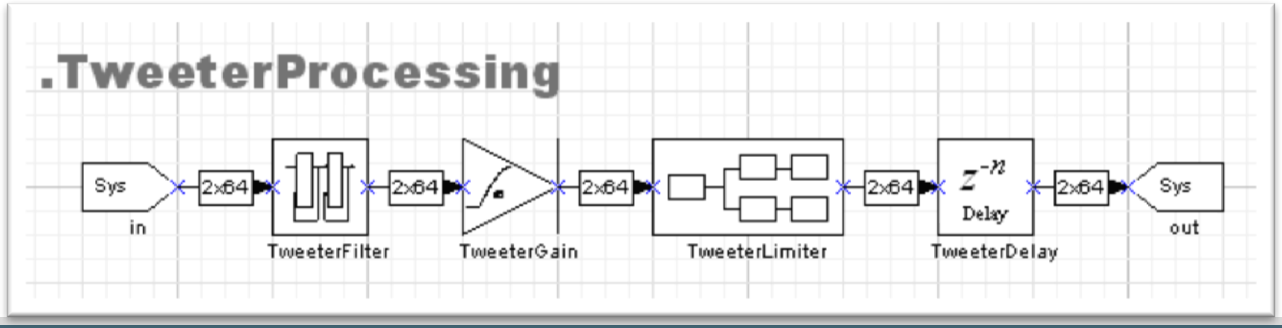
Signal Flow



Top-level system



Bass subsystem



Tweeter subsystem

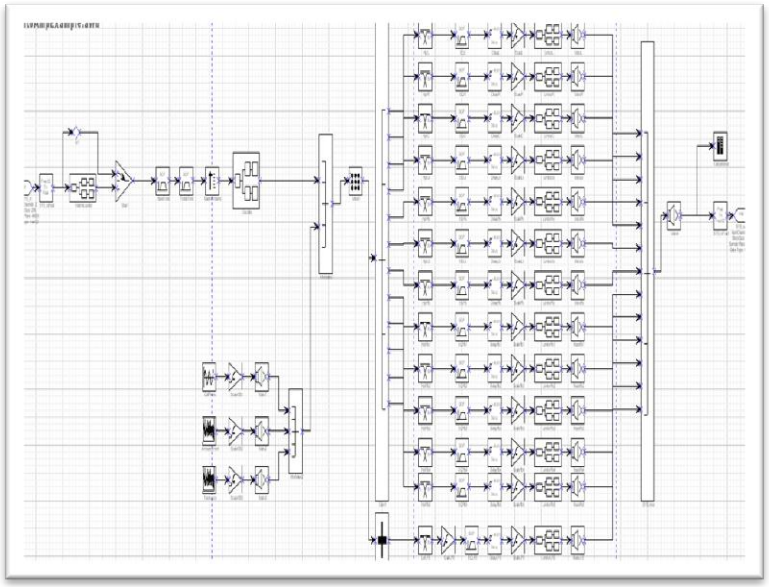
Loudspeaker Processing Results

	DSP			Cortex-M4	
Module Name	**MIPS**	**SIMD**		**MIPs**	**SIMD**
SYS_toFloat	0.28	No		1.25	N/A
BassTone	0.59	Yes		2.72	N/A
TrebleTone	0.59	Yes		2.09	N/A
VolumeControl	0.68	N/A		2.79	N/A
Crossover	2.75	Yes		10.41	N/A
BassProcessing.BassAdder	0.72	Yes		2.02	N/A
BassProcessing.BassFilt	1.54	No		5.78	N/A
BassProcessing.BassGain	0.49	No		1.35	N/A
BassProcessing.BassLimiter	3.03	N/A		12.79	N/A
BassProcessing.BassDelay	0.3	N/A		1.41	N/A
BassProcessing.Interleave1	0.52	N/A		0.74	N/A
TweeterProcessing.TweeterFilter	1.65	Yes		11.87	N/A
TweeterProcessing.TweeterGain	0.45	Yes		2.14	N/A
TweeterProcessing.TweeterLimiter	6.87	N/A		27.42	N/A
TweeterProcessing.TweeterDelay	0.52	N/A		2.28	N/A
ListenMux	0.62	N/A		1.52	N/A
SYS_toFract	0.29	No		3.45	N/A
Total MIPs	**21.89**			**92.03**	

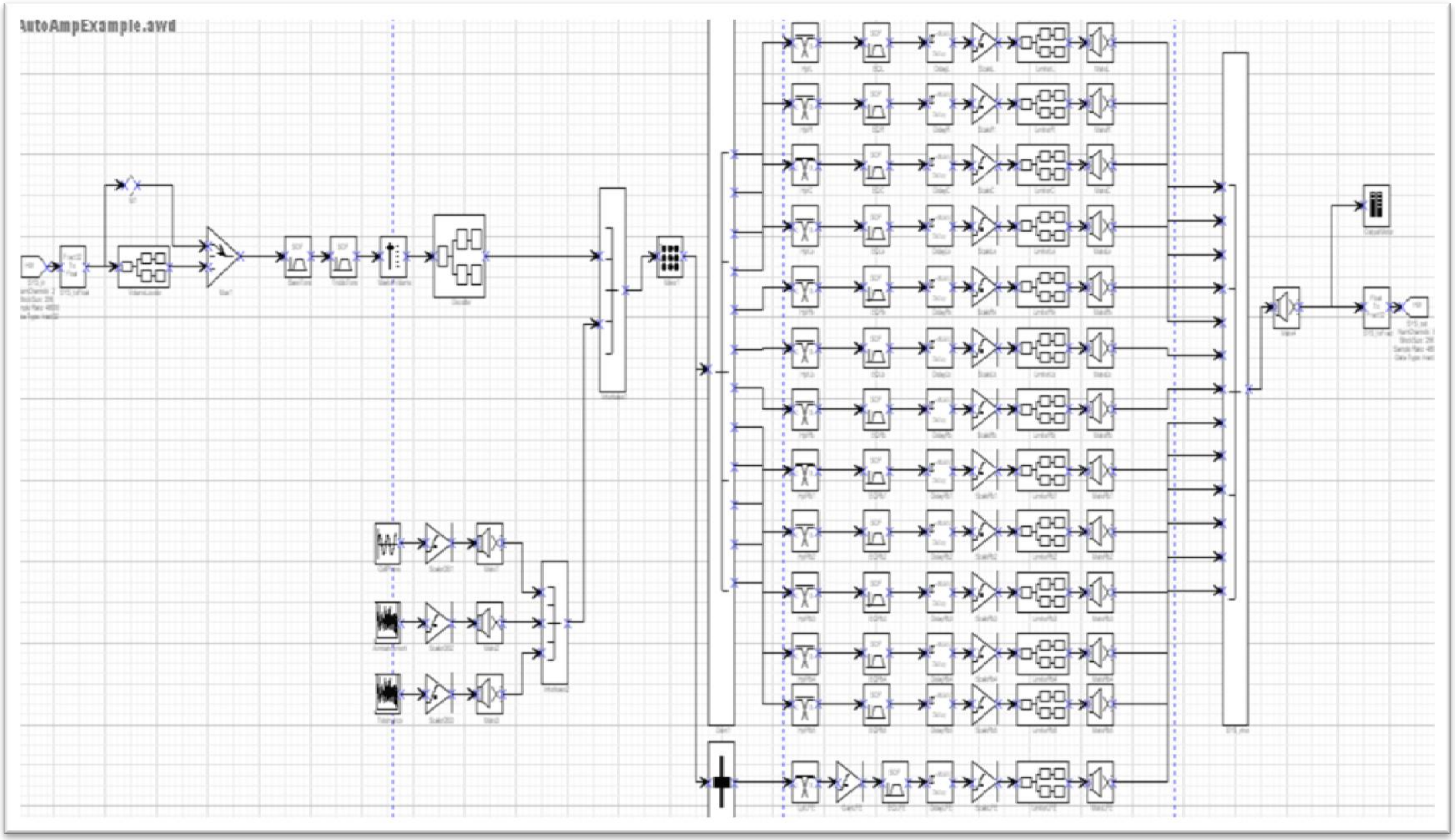
Premium Automotive System

- 16 input and 13 output channels
- 10 band graphic equalizer
- Spectrum analyzer
- Volume control with Fletcher-Munson compensation
- 6 announcement channels with signal dependent ducking
- Speed dependent equalization and volume control
- Over 165 Biquads for loudspeaker equalization
- Compressors, limiters, and delays on all loudspeaker channels.
- Test signal generation for in car diagnostics
- Over 300 individual audio modules!

Representative of a production automotive audio system.



Automotive Signal Flow



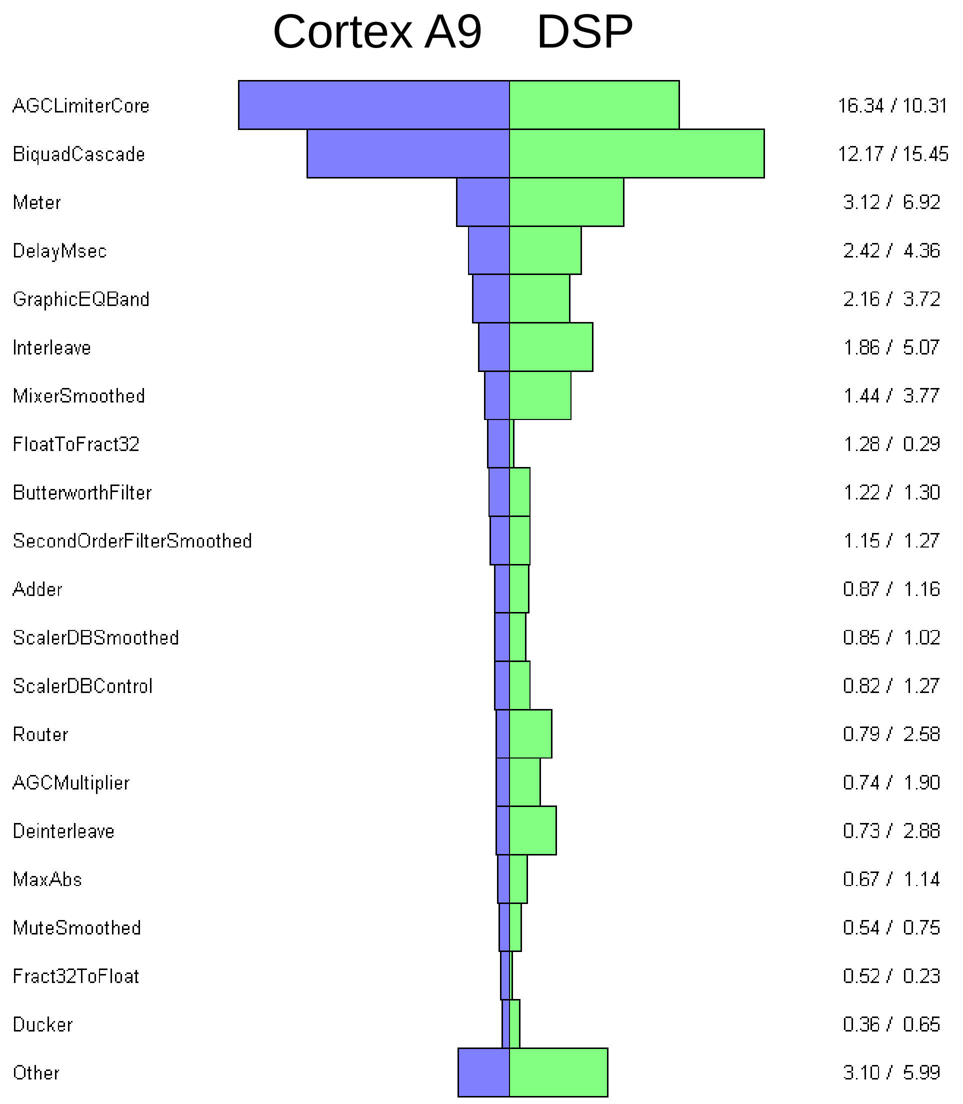
Automotive Benchmarking Results

<u>DSP</u>

- 32 sample block size
- 72% = 288 MHz

<u>Cortex-A9</u>

- 256 sample block size
- 53% = 530 MHz



Conclusion

- DSPs and microcontrollers are on a collision course
- Through careful programming techniques you can significantly increase the processing throughput of microcontrollers
  - Digital signal controllers (e.g., Cortex-M4) are capable of entry-level 2 channel audio processing
  - High-end application processors (e.g., Cortex-A8 / A9) are capable of multichannel premium audio processing
- Prediction
  - Microcontrollers will continue to add specialized audio peripherals in order to gain a foothold in the market
  - DSPs will be pushed out of high volume consumer sockets and be reserved for specialized applications

Thank You!