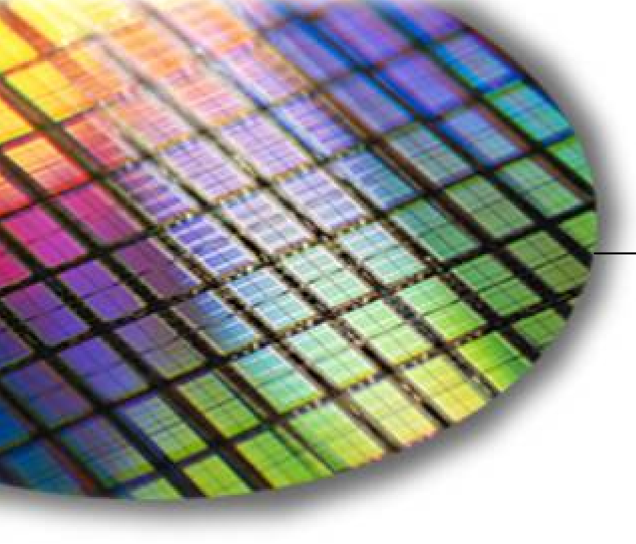


The World Leader in High Performance Signal Processing Solutions

Blackfin® Optimizations for Performance and Power Consumption

Presented by:

Merril Weiner
Senior DSP Engineer

About This Module

This module describes the different ways to optimize the software design to increase performance on the Blackfin processor and/or reduce power consumption for low power applications.

- **Example implementations for audio players and video players will be used to explain the concepts in this module.**
- **Demo implementation of an audio and video player using these concepts will be given at the end of this presentation, showing the performance and power.**
- **It is recommended that viewers have seen the module Basics of Building a Blackfin Application and have a working knowledge of the Blackfin processor.**

Module Outline

1. **Overview**
2. **Optimizing Internal Memory Use**
3. **Optimizing External Memory Use**
4. **Power Modes**
5. **Demos**

Overview

Strategies

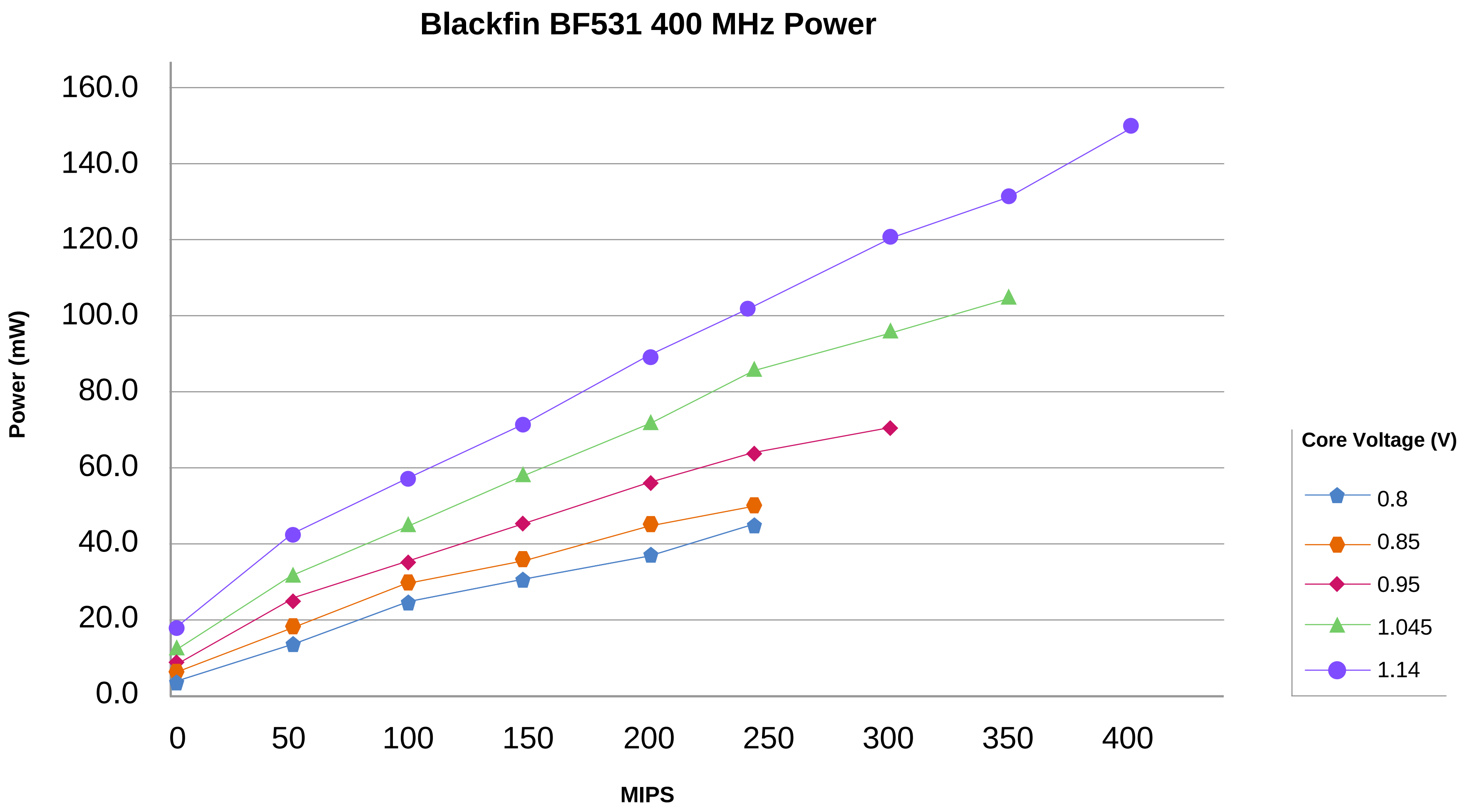
Optimized Software Design = Increased Performance
= More Features or Lower Power Consumption

- Optimized software design means lower MIPS required, allowing for additional features to be added or to reduce the frequency (less dynamic power) and voltage (less static power).
- Optimized code is more than just turning on the optimization switch on the compiler.

How can a system be further optimized?

- Optimize use of internal memory (L1 instruction and L1 data).
- Optimize use of external memory (L3-SDRAM).
- Effective use of power modes.
- Effective use of Blackfin and SDRAM settings.

More Features or Lower Power



Blackfin BF531 Data Sheet Rev. D and Application Note EE-229

Note: Full on, typical at 25C

Software Example—Portable Media Player

Example implementation for the Blackfin family of a low power portable media player

- **Audio-only playback example**
  - Blackfin BF531
- **Audio/video playback example**
  - Blackfin BF533
- **These techniques are applicable across all Blackfin processors**

Optimizing Internal Memory Use

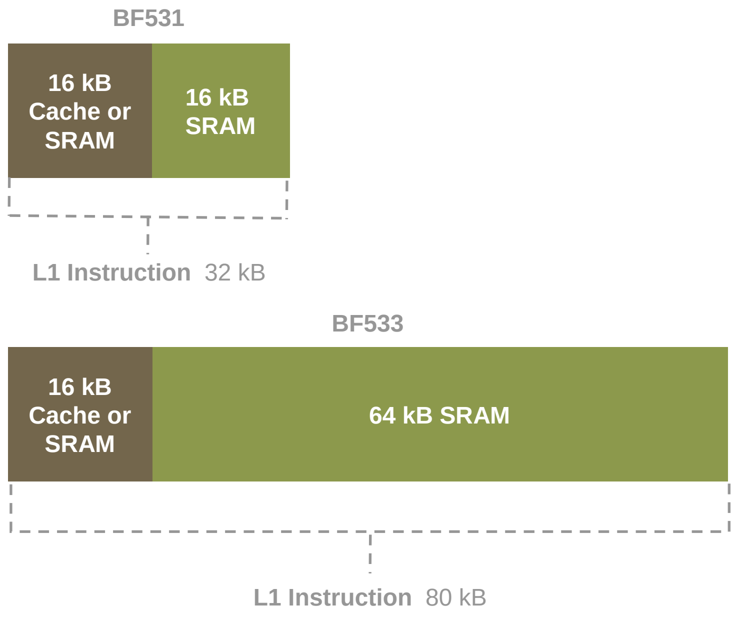
L1 Instruction SRAM vs. I-Cache

When algorithms exceed the 16 kB of L1 I-cache, using I-cache may not be sufficient for optimal performance. In these cases, use L1 instruction SRAM for overlays.

- **Audio decoders < 16 kB instruction**
- **Video decoders > 16 kB instruction**

Using L1 instruction SRAM vs. L1 I-cache

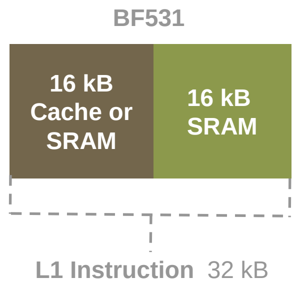
- **Cache-only model**
  - Turn on I-cache
- **Overlay model**
  - Turn off I-cache
  - Use all of L1 instruction as SRAM
- **Hybrid model**
  - Turn on I-cache
  - Use remaining L1 instruction as SRAM



L1 Instruction SRAM vs. I-Cache (cont.)

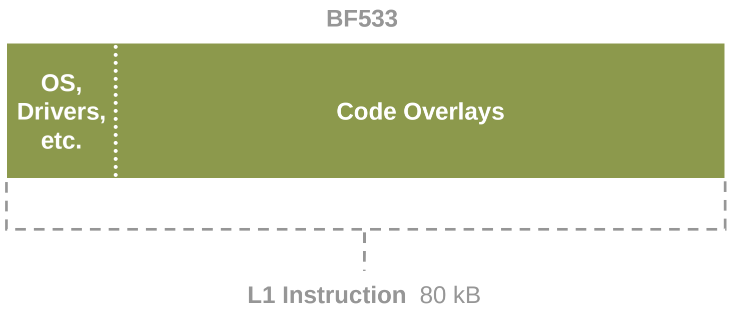
Using L1 instruction SRAM vs. L1 I-cache

- **Cache-only model**
  - **Only use L1 instruction for I-cache.**
  - **Pros**
    - Perfect for audio-only applications or where all critical modules have < 16 kB code.
    - Works well with multiple critical modules.
    - Ease of programming.
    - Allows for buying a processor with less L1 instruction (e.g., Blackfin BF531 instead of Blackfin BF533).
  - **Cons**
    - May have significantly worse performance when critical modules > 16 kB code.
  - **Summary**
    - Good for most cases, especially when critical modules have < 16 kB code space each.
    - Only option for µClinux™.



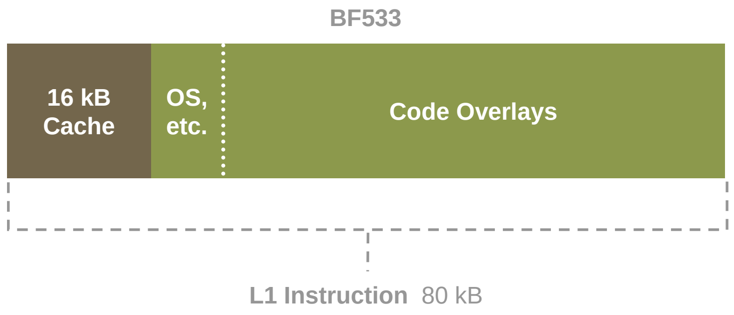
L1 Instruction SRAM vs. I-Cache (cont.)

- **Overlay model**
  - **Only use L1 instruction for code overlay.**
  - **Multiple overlays or single overlay.**
  - **Pros**
    - Increased performance for code placed in L1 instruction SRAM as code execution may be delayed if it is not currently in the cache.
  - **Cons**
    - Decreased performance for code that does not fit in L1 instruction SRAM must be placed in L3.
  - **Issues**
    - If all critical code can fit in L1 instruction SRAM, do not use overlays but place code directly in L1 instruction SRAM.
    - If multiple critical modules cannot all fit in L1 instruction SRAM, use code overlays and swap the modules in as needed.
    - Noncritical modules are not worth the context switch (swapping into L1 instruction SRAM) and must be left in L3.
    - Need to manage multiple overlays.
    - PGO Linker tool available for help with sections and overlays.
  - **Summary**
    - Optimal if all critical modules fit in the code overlay space and there is very little or no code executing out of L3.



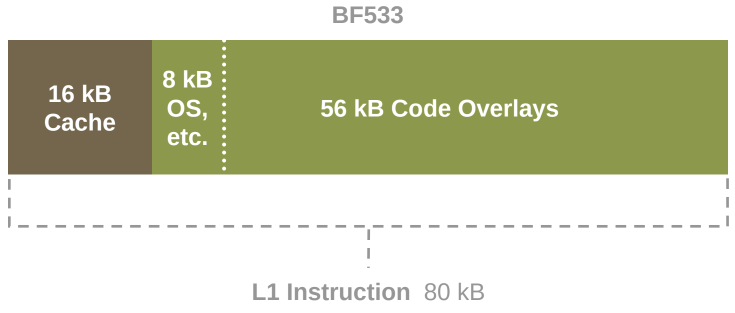
L1 Instruction SRAM vs. I-Cache (cont.)

- **Hybrid model**
  - **Use L1 instruction SRAM and I-cache.**
  - **Place as much critical code as possible in L1 instruction SRAM overlays.**
  - **Pros**
    - Excellent performance for critical modules placed in L1 instruction SRAM overlays.
    - Very good performance for noncritical modules that use I-cache.
  - **Cons**
    - Requires a processor with more than 32 kB of L1 instruction SRAM.
  - **Issues**
    - Same as overlay model.
  - **Summary**
    - Optimal performance for multiple critical modules > 16 kB and multiple noncritical modules executing out of L3.



L1 Instruction SRAM vs. I-Cache (cont.)

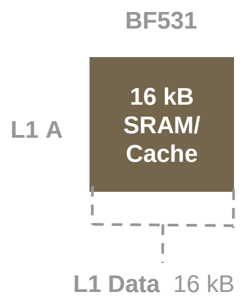
- **Video example**
  - **Use hybrid model, the best of both worlds.**
  - **Memory map**
    - Turn on I-cache (16 kB of L1).
    - Set aside L1 instruction SRAM for OS and drivers (8 kB).
    - Remaining L1 instruction SRAM used for code overlays (56 kB).
  - **Performance**
    - Cache-only model—20% performance degradation due to large critical modules (> 64 kB code).
    - Overlay model—10% performance degradation due to huge performance degradation of noncritical modules (10–50x) executing out of L3.
    - Hybrid model—best performance!



L1 Data SRAM vs. D-Cache

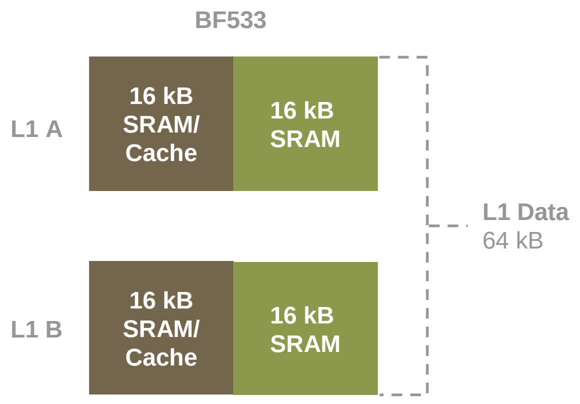
When data exceed 16 or 32 kB, using L1 D-cache may not be sufficient for optimal performance.

- **Allocate memory for each type of data:**
  - Read-only data (r)—static variables (e.g., tables)
  - Read/write data (r/w)—state variables
  - Temp data (temp)—temporary variables (e.g., internal buffers)
- **Audio decoders < 16 kB data**
- **Video decoders > 32 kB data**
- **Video post-processing > 32 kB data**



Using L1 data SRAM vs. L1 D-cache

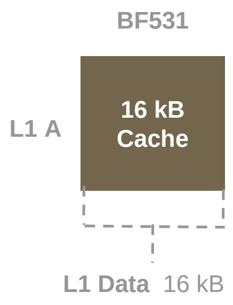
- **Cache-only model**
  - Turn on D-cache
- **Overlay model**
  - Turn off D-cache
  - Use all L1 data as SRAM
- **Hybrid model**
  - Turn on D-cache
  - Use remaining L1 data as SRAM



L1 Data SRAM vs. D-Cache (cont.)

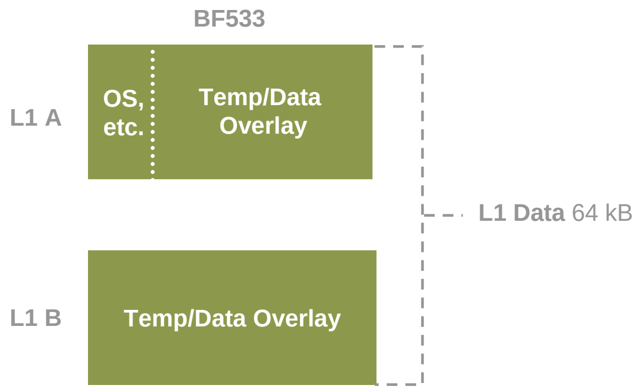
Using L1 data SRAM vs. L1 D-cache

- **Cache-only model**
  - **Only use L1 data for D-cache.**
  - **Data placement**
    - All variables (r, r/w, temp) are placed in L3 and let D-cache do its job.
  - **Pros**
    - Perfect for audio-only applications or where each critical module uses < the data cache size.
    - Works well with multiple critical modules.
    - Ease of programming.
    - Allows for buying a processor with less L1 data (e.g., Blackfin BF531 instead of Blackfin BF533).
  - **Cons**
    - May have significantly worse performance when each critical module uses > the data cache size.
  - **Summary**
    - Good for most cases, especially when each critical module uses < the data cache size.
    - Only option for µClinux.



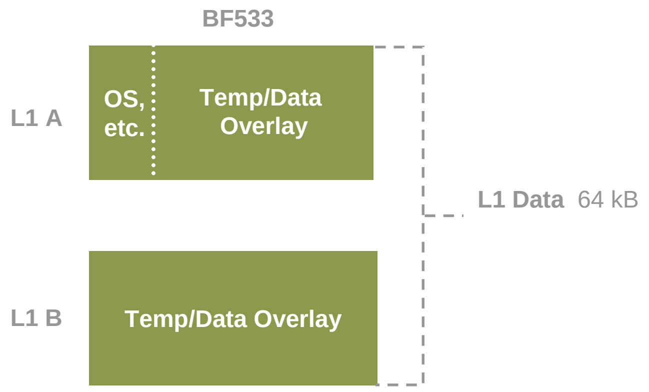
L1 Data SRAM vs. D-Cache (cont.)

- **Overlay model**
  - **L1 D-cache is turned off.**
  - **Data placement:**
    - Place state variables (r/w) and static variables (r) in L1 data SRAM overlays.
    - Place temp variables (temp) in L1 data SRAM.
    - Place concurrent buffers in separate 4 kB L1 data SRAM banks.
  - **Multiple overlays or single overlay**
  - **Pros**
    - Increased performance for modules that use L1 data overlays.
  - **Cons**
    - Decreased performance for modules where data must be accessed directly from L3.



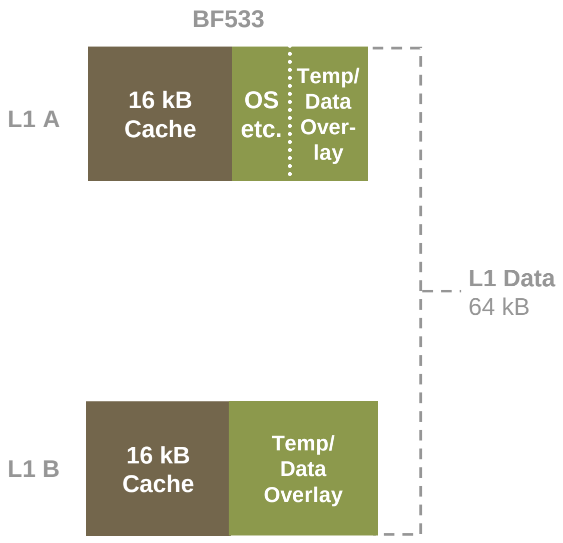
L1 Data SRAM vs. D-Cache (cont.)

- **Overlay model (cont.)**
  - **Issues**
    - If all data can fit in L1 data SRAM, do not use overlays but place data directly in L1 data SRAM.
    - If multiple critical modules have a total data size > 64 kB and cannot fit in L1 data SRAM, use memory overlays to swap the modules' data in and out as needed.
    - Noncritical modules are not worth the context switch (swapping data into L1 data SRAM) and must leave the data in L3.
    - Need to manage multiple overlays.
    - PGO Linker tool available for help with sections and overlays.
  - **Summary**
    - Optimal if the data for critical modules fits in the overlay and if noncritical modules access very little or no L3 data.



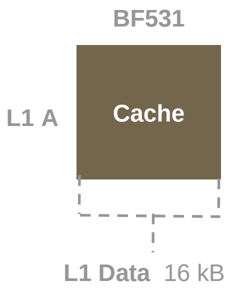
L1 Data SRAM vs. D-Cache

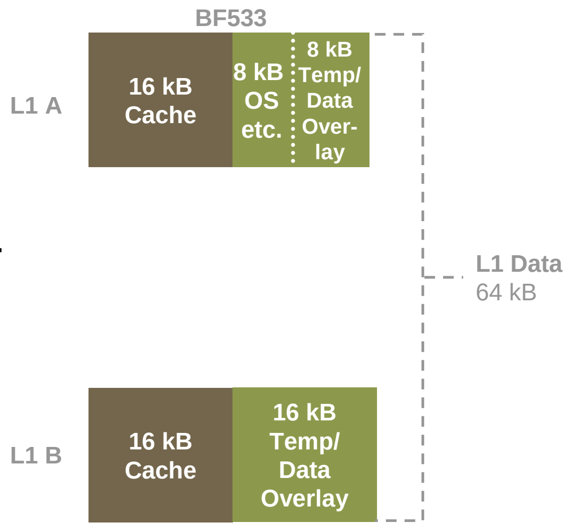
- **Hybrid model**
  - **L1 D-cache is turned on in write back mode.**
  - **Use remaining L1 data for SRAM.**
  - **Data Placement**
    - Place state variables (r/w) in L3 and let D-cache do its job.
    - Place static variables such as critical tables (r) in L1 data SRAM overlays.
    - Place temp variables (temp) in L1 data SRAM.
    - Place concurrent buffers in separate 4 kB L1 data SRAM banks.
  - **Pros**
    - Excellent performance for critical modules.
    - Very good performance for noncritical modules that use D-cache only.
  - **Cons**
    - Requires a processor with more than 32 kB of L1 data.
  - **Issues**
    - Same as overlay model.
  - **Summary**
    - Optimal performance for multiple critical modules that use > 32 kB data and multiple noncritical modules.



L1 Data SRAM vs. D-Cache (cont.)

- **Audio example**
  - **Use cache model.**
  - **Memory map**
    - All data is in L3 and D-cache does its job.
- **Video example**
  - **Requires Blackfin processors with 64 kB of L1 data.**
  - **Use hybrid model, the best of both worlds.**
  - **Memory map**
    - L1A—16 kB data cache, 8 kB OS and drivers, 8 kB temp/overlay.
    - L1B—16 kB data cache, 16 kB temp/overlay.
  - **Performance**
    - Cache-only model has ~100% system performance degradation.
    - Overlay model is 7% better for critical modules, but noncritical modules are much worse for a net ~20% system performance degradation.
    - Hybrid model—best performance!





Cache and DMAs

Data coherency—outgoing data

- Cache is not automatically flushed.
- Data coherency issue can exist if peripheral DMAs read from SDRAM where the data is cached and not flushed.
- Cached buffers must either be flushed before starting a peripheral DMA or be marked as non-cached in the CPLB tables.

Data coherency—incoming data

- Cached data is not automatically updated.
- Data coherency issue can exist if peripheral DMAs write to SDRAM where the data is cached and not invalidated.
- Cached buffers must either be invalidated when peripheral DMA updates data or mark data buffers as non-cached in the CPLB tables.

Memory Pipelining

MemDMA to bring data into L1 data SRAM instead of using the D-cache

- **Use temp data in L1 data SRAM for I/O buffers.**
- **Use memory DMAs to asynchronously transfer I/O data.**
- **Execute code while DMAs are active.**
- **Fetch next set of data, store last set of data while working on the current set of data.**
- **"Pipelining" reduces MIPS by up to 50%.**

Video example

- **Video post-processing is memory bandwidth limited.**
- **While working on line n, fetch line n+1 and store line n-1.**
- **Pipelining reduces MIPS by up to ~40%.**

L1 Data Scratchpad

L1 data scratchpad is 4 kB for all processors.

- **Cannot be used for D-cache.**
- **Cannot be accessed with DMAs.**
- **Best use is stack.**

For non-OS environments or single task solutions, place stack in the scratchpad.

For RTOS solutions, each task must have its own stack.

- **Place most critical tasks' stacks or larger stacks in scratchpad.**
- **Place remaining stacks in L3, allowing for data cache to bring it into L1.**
- **Experiment.**

Video example

- **Use VisualDSP++ Kernel (VDK).**
- **Experimented with which stacks are placed in the L1 scratchpad ➔ Saved ~20 MIPS.**

Optimizing External Memory Use

Memory Settings

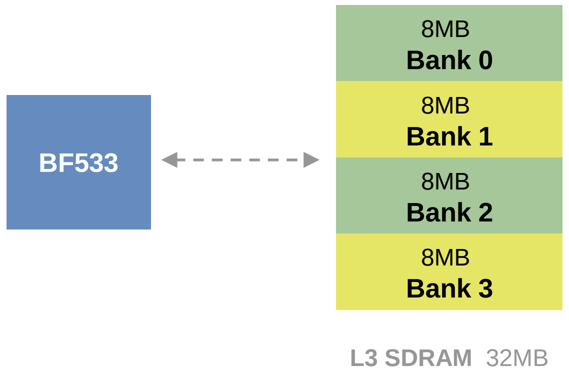
It is important to set memory priorities correctly.

- **In most Blackfin processors, the peripheral DMAs can be set to have higher priority than core access to L3 (instruction and data). This is critical to avoid peripheral FIFO underruns/overruns.**
- **Some Blackfin processors allow for even greater control of DMA priorities.**

External Memory Banks

Each Blackfin supports at least 1 external memory bank of SDRAM.

- **Each external SDRAM banks consists of 4 sub-banks.**
- **At any time, only 1 page is open in each of the 4 sub-banks.**
  - Accesses to open pages improve performance.
  - Accesses to closed pages require closing an open page and opening a new page (multiple SCLK latency).
- **Random accesses to the same SDRAM sub-bank more than a page-size apart reduce performance.**
- **Place concurrently accessed buffers in separate SDRAM sub-banks whenever possible.**
- **For Mobile SDRAM sub-banks, unused banks can be placed in self-refresh mode to save power.**



External Memory Banks (cont.)

Audio example (Mobile SDRAM)

- **Place all buffers in one bank.**
- **Place other banks in self-refresh to save on power.**

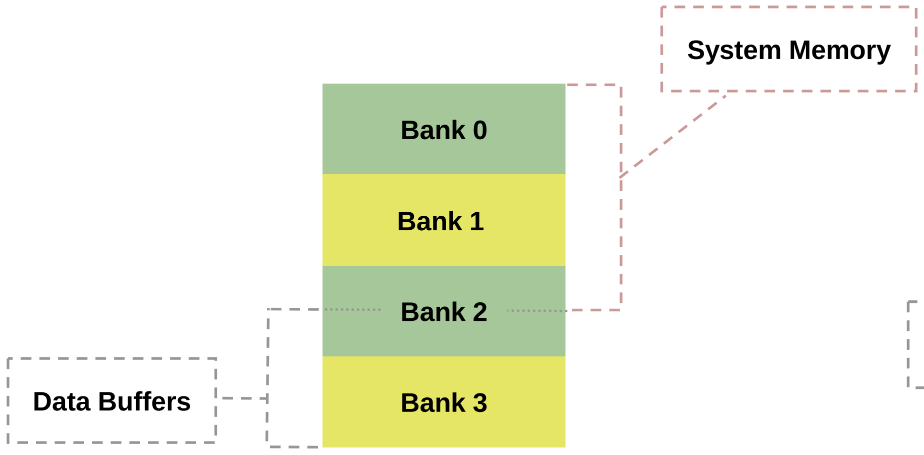
Video example

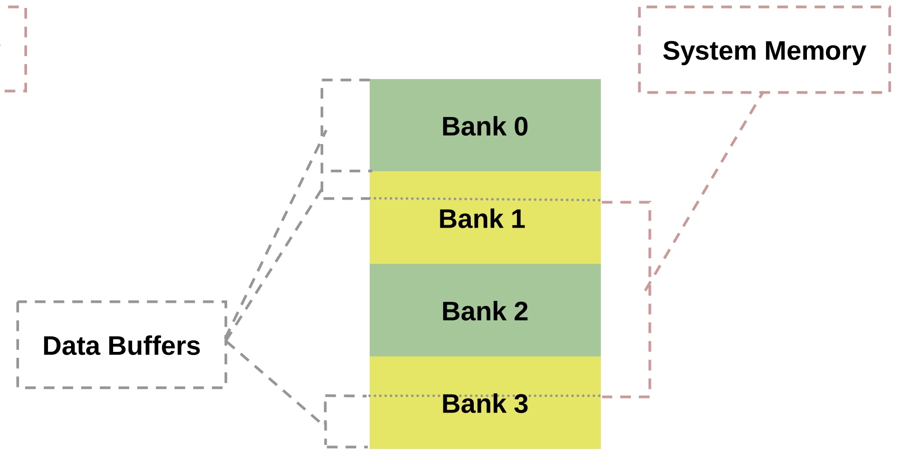
- **For video decoders/encoders**
  - Place two input reference frames in separate banks.
  - Place output frame in one of the remaining banks.
  - Place audio buffers in fourth bank.
  - When banks are not set up correctly, performance degrades 100% due to page misses.

- **Be careful when changing the physical SDRAM size as the memory map across banks will change.**

OS and Contiguous Memory

Some RTOS's and µClinux require that the memory used by the OS be a single contiguous block. With these OS's, the bank of a data buffer cannot be guaranteed. The following diagram shows how to use three banks of SDRAM for data buffers while leaving over half of the memory for the OS.





Power Modes

Hibernate and Sleep Modes

Hibernate mode

- **Best power savings for non-real-time systems.**
- **This mode should be used for long idle periods when peripherals are not used continuously.**
- **Core power and core clock are turned off.**
- **L1 memory and registers are not maintained.**
- **Consumes only ~50 uA of current by saving core static and dynamic power.**
- **Wake up on RTC or user input and takes ~600 us (300 us h/w + 300 us s/w).**

Sleep mode

- **Best power savings for real-time systems.**
- **This mode should be used within the RTOS or system idle routine for shorter idle periods or when peripherals are used continuously.**
- **Core power is kept on, and core clock is turned off.**
- **Peripherals continue to run; L1 memory and all registers are maintained.**
- **Saves dynamic power.**
- **Wake up configurable on any interrupt and takes 10 CCLK cycles (40 ns at 250 MHz, 185 ns at 54 MHz).**

Hibernate and Sleep Modes (cont.)

Deep sleep mode

- Specific peripheral interrupts including the RTC can wake up the processor.
- Not as convenient as sleep mode and power consumption is not much.

Frequency and Voltage

In order to conserve power on the Blackfin, be aware of the different tiers of speed and voltage. Here’s an example:

<= 250 MHz	0.80 V
<= 280 MHz	0.85 V
<= 333 MHz	0.95 V
<= 363 MHz	1.045 V
<= 400 MHz	1.14 V

(refer to your processor datasheet)

The static current on a Blackfin is directly related to voltage and temperature.

- **To reduce static power, set the lowest voltage that allows for the necessary MIPS for the current application.**

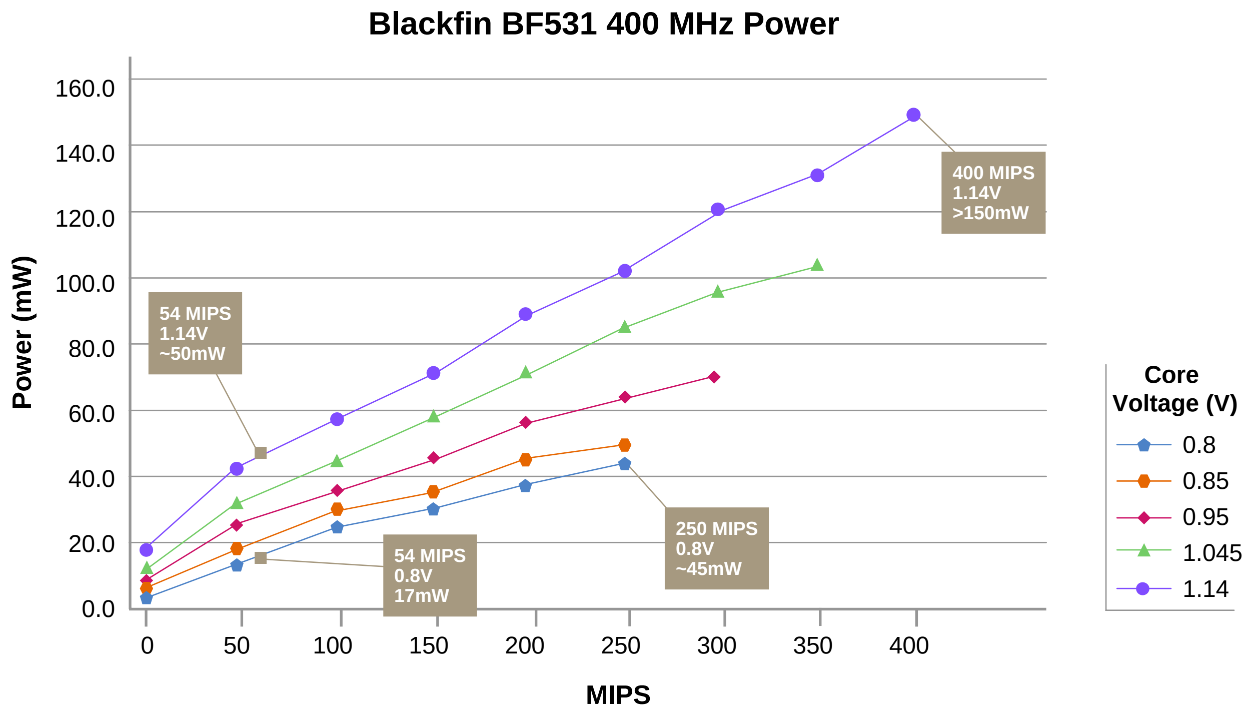
The dynamic current on a Blackfin is directly related to clock cycles per second.

- **Set the frequency to the maximum speed allowed by the voltage level.**
- **Sleep when idle to reduce the dynamic power.**

Frequency and Voltage Audio

Audio example (Blackfin BF531 400 MHz)

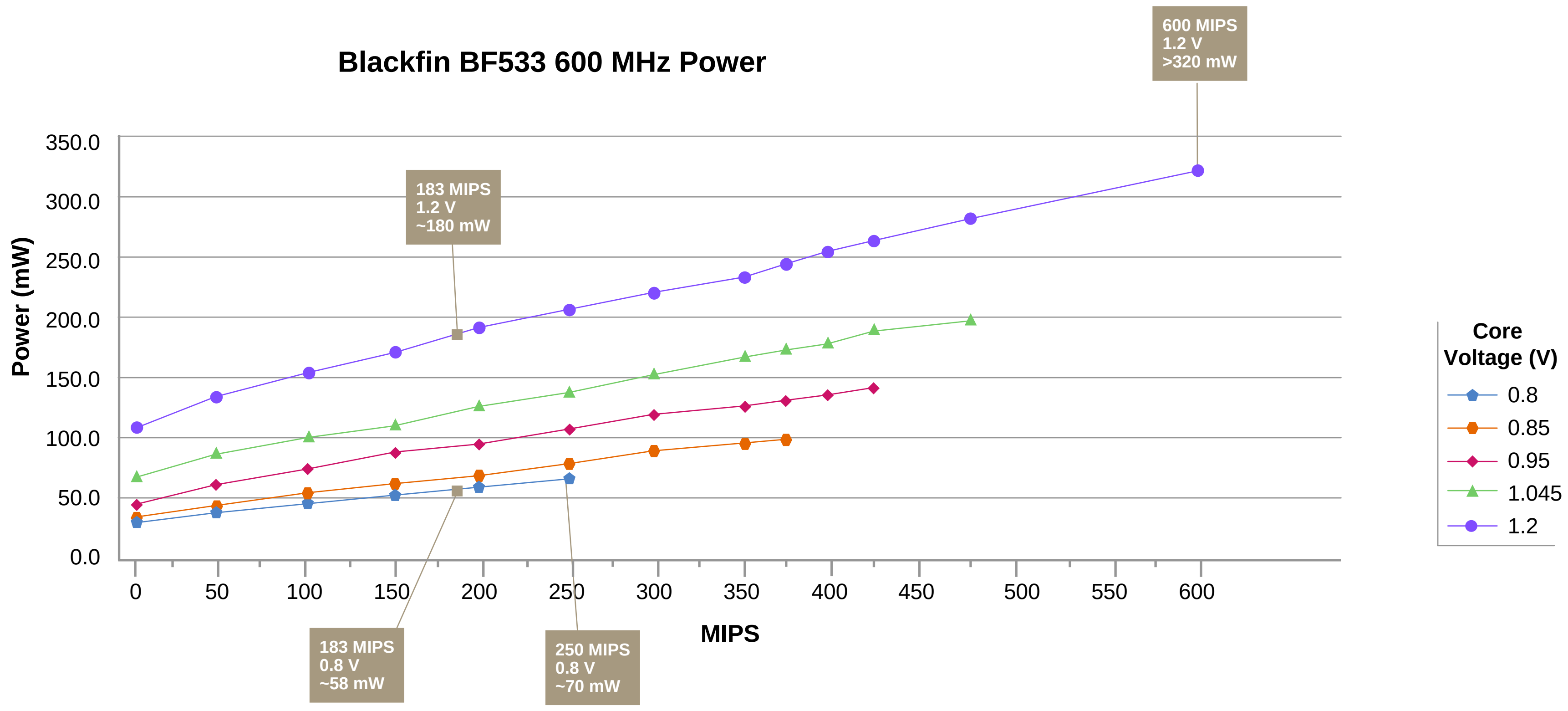
- Audio decode plus post-processing consumes 54 MIPS.
- Set voltage = 0.8 V, Core clock = 243 MHz.
- Sleep 78% of the time.
- Audio DMA frame completion interrupts wake-up processor.
- Consumes ~17 mW core (dynamic + static) at 25C.



Frequency and Voltage Video 1 QVGA

Video example 1 (Blackfin BF533 600 MHz)

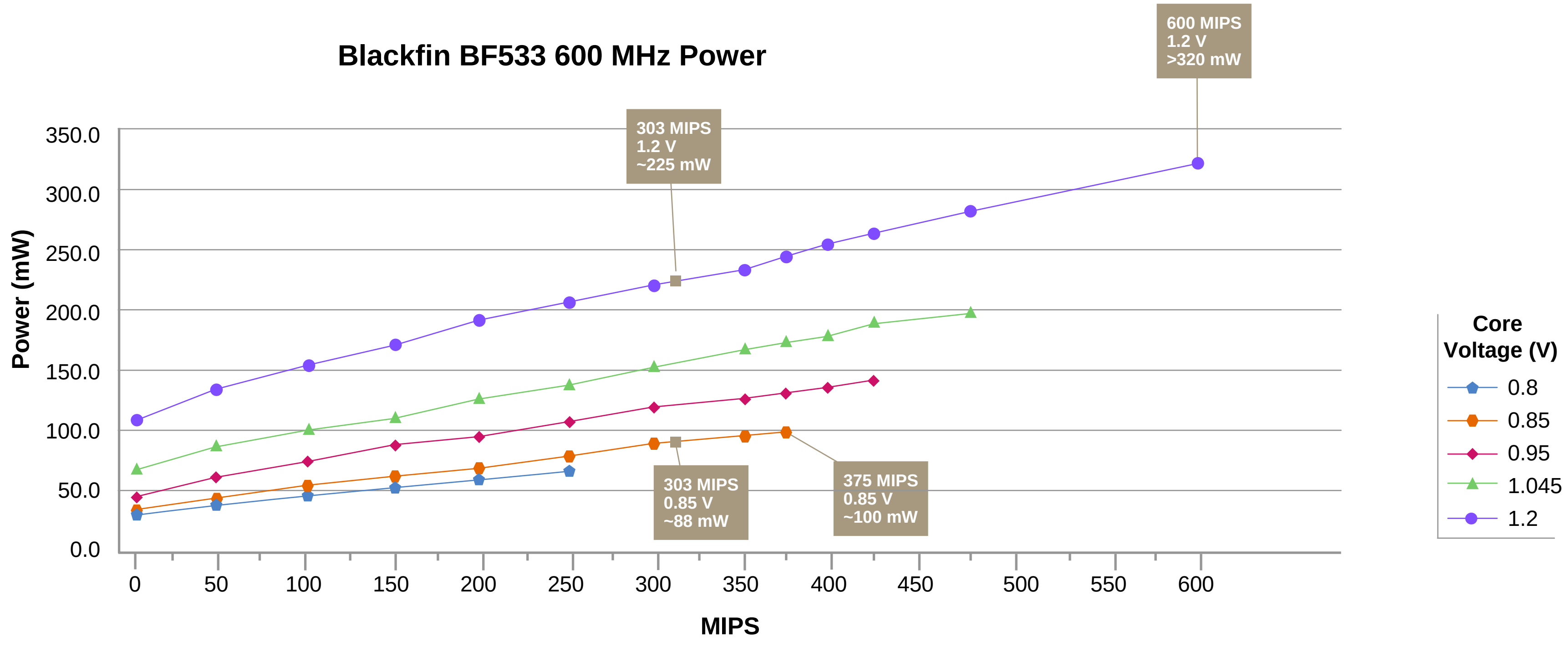
- QVGA decode example consumes 183 MIPS.
- Set voltage = 0.8 V, Core clock = 243 MHz.
- Sleep 25% of the time.
- Consumes ~58 mW core (dynamic + static) at 25C.



Frequency and Voltage Video 2

Video example 2 (Blackfin BF533 600 MHz)

- **WQVGA (480 x 270) decode example with graphics consumes 303 MIPS.**
- **Set voltage = 0.85 V, Core clock = 351 MHz.**
- **Sleep 14% of the time.**
- **Consumes ~88 mW core (dynamic + static) at 25C.**



System Power and External Memory Speed

Reducing external memory speed reduces system power consumption.

- **System performance degrades when memory accesses exceed half total memory bandwidth.**
- **Calculate number of memory accesses per second in order to calculate minimum memory bandwidth.**
- **Adjust system clock so that total memory bandwidth is twice necessary memory accesses per second.**
- **May need to reduce the VCO to lower the system clock (SCLK) sufficiently to save on external memory power consumption.**
  - Since modifying the VCO affects the CCLK, ensure that the CCLK is fast enough to execute applications.

Audio example

- **Memory accesses**

  = 96 kbps + 48 kHz samples x 4 bytes/sample x 4 (decode and output + post-proc)

  = 0.4 million accesses per second

  ➔ 0.8 MHz minimum
- **CCLK = 243 MHz, could set PLL_DIV to the max ➔SCLK 8.1 MHz.**
- **To lower memory power consumption further, lower VCO to 54 MHz. CCLK = 54 MHz and set SCLK to about 2.7 MHz.**

System Power and External Memory Speed (cont.)

Balance reduction of memory speed against core performance for lowest system power consumption.

- **Slowing down memory speed can cause MIPS to increase.**
- **Make sure that MIPS does not exceed MHz.**
- **Balance power savings for memory against power loss for core.**
- **Optimize memory use.**
  - When pipelining code execution with DMA memory transfers, use DMA frame completion interrupts and sleep while waiting for DMA to complete saving on dynamic power.
  - If the interrupt overhead is too great, then polling is necessary. Slow down CCLK to reduce the amount of time spent polling for DMAs to complete saving on dynamic power.
- **Experiment.**

System Power and External Memory Speed (cont.)

Video example

- **QVGA decode at 500 kbps**
- **Memory accesses**
  - = 500 kbps + 320 x 240 x 12 bpp x 3 ref frames x 30 fps (decode) + 320 x 240 x (12 bpp + 18 bpp) x 30 fps (post-proc) + 320 x 240 x 3 bpp x 60 fps (display)
  - = 17 million accesses per second
  - ➔ 35 MHz minimum
- **If VCO = CCLK = 243 MHz, then PLL_DIV = 0x0007 and SCLK = ~35 MHz**
- **Due to memory bandwidth limited nature of some modules, SCLK = 35 MHz as opposed to SCLK > 100 MHz will cause DMAs to take 3x as long and MIPS will increase 200%!**
- **Due to a high number of interrupts, more efficient to poll for interrupts to complete.**
- **Setting PLL_DIV = 0x4 is optimum for core dynamic power while PLL_DIV = 0x7 is optimum for memory power consumption. Best solution is PLL_DIV = 0x5 (default value).**
- **Reduce CCLK: Set VCO = CCLK = 216 MHz, PLL_DIV = 0x5 ➔ SCLK = 43.2 MHz**
- **Experiment**

External Memory Banks for Mobile SDRAM

Memory power consumption is directly related to the number of active banks.

- **For example, with 1.8 V Mobile SDRAM, one bank consumes ~1 mW in self-refresh mode as opposed to ~20 mW in active mode.**
- **Place as many banks in self-refresh as possible for the task at hand.**
- **Balance power consumed by MIPS against power consumed by memory banks.**
  - 1 bank = ~20 mW = ~85 MIPS (for 0.8–0.85 V)
  - If reducing by one bank causes a performance degradation < 85 MIPS, place the bank in self-refresh.
  - Blackfin processors allow one, two, or four active banks.
  - May place two or three banks in self-refresh for power savings.

External Memory Banks for Mobile SDRAM (cont.)

Audio example

- Audio decode + audio post-processing example = 54 MIPS with one bank
- Keep one bank active and place three banks in self-refresh for a net savings of ~60 mW system wide!

Video example

- QVGA video example = 183 MIPS with four banks
- Video solution with 3 banks suffers 22 MIPS degradation < 85 MIPS. Definitely worth it!
- Video solution with 2 banks suffers 212 MIPS degradation > 170 MIPS. Not worth it.
- Keeping three banks active is not an option, and keeping only two banks active is not worth it.
- Keep all four banks active.

Conclusion

To fully optimize your system for increased performance and/or reduced power consumption, it requires a diligent look at:

- **Voltage and frequency settings**
- **L1 settings and use**
- **L3 settings and use**
- **Power modes**
- **Experiment**

Blackfin processor architecture is filled with many ways to increase performance beyond initial expectations.

For Additional Information

Analog Devices website which has links to white papers, manuals, data sheets, FAQs, Knowledge Base, sample code, development tools and much more:

- **www.analog.com/blackfin**

For specific questions click on the "Ask a question" button.

Demos

Demos

Example implementation for the Blackfin family of a low power portable media player

- **BF533 EZ-KIT Lite®**

Audio only example

- **AAC-LC audio decode**
- **Audio sample rate converter**
- **54 MHz, 0.8 V, ~39 mW (~17 mW = BF531) core power for audio-only playback**

Audio/Video example

- **QVGA MPG4 ASP**
- **Video post-processing with alpha graphics blending**
- **AAC-LC audio decode**
- **Audio sample rate converter**
- **243 MHz, 0.8 V, ~70 mW core power for audio/video playback**

This software was created for demonstration purposes only and is not available for distribution.