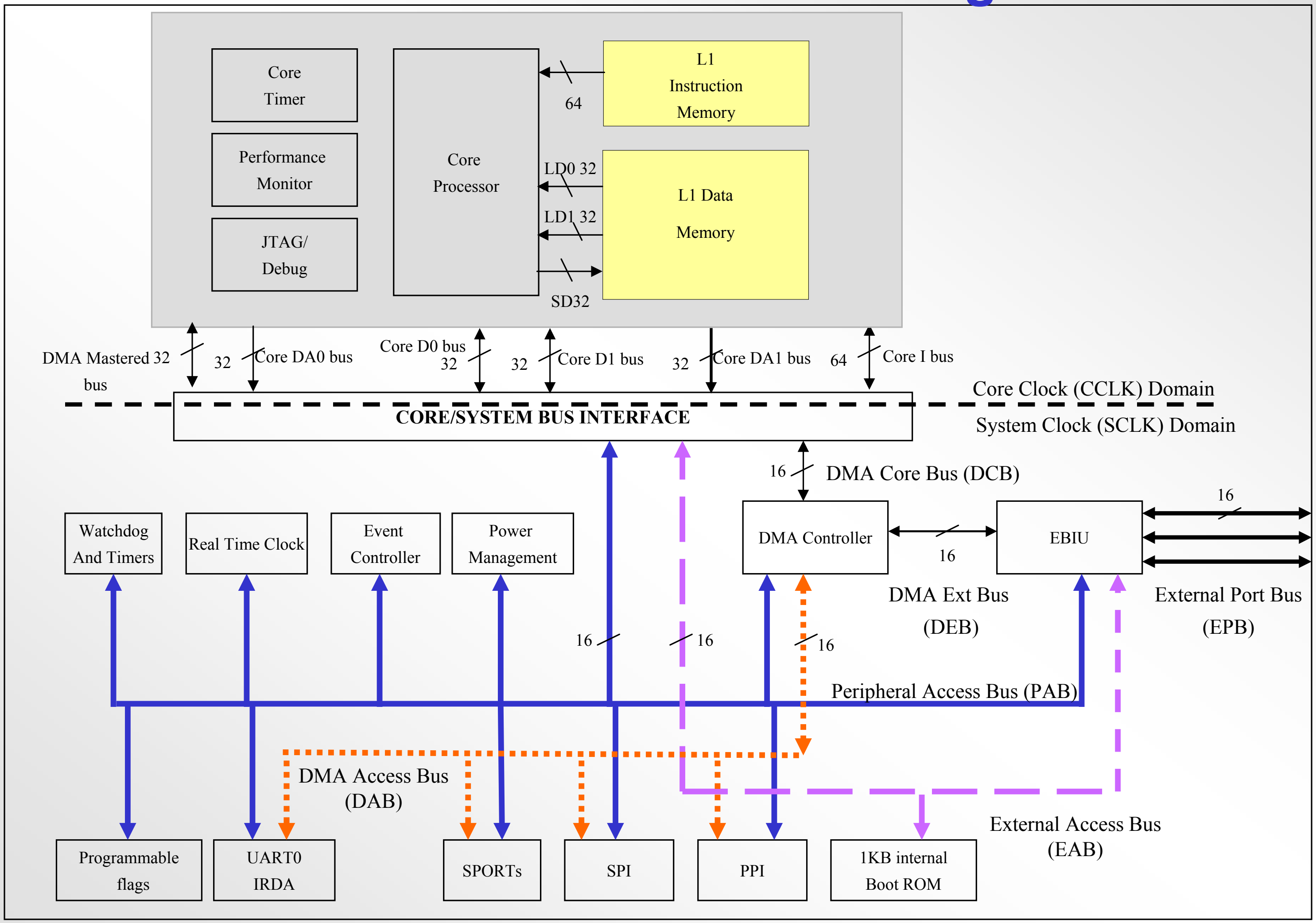
Section 6

Blackfin ADSP-BF533 Memory

ADSP-BF533 Block Diagram



KAZTEK ENGINEERING

ANALOG DEVICES

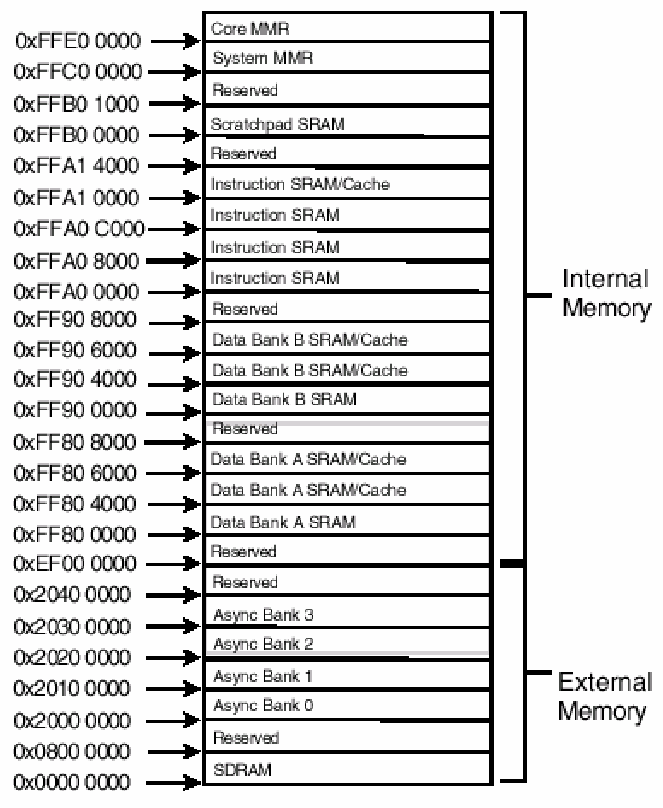
Blackfin Internal SRAM

ADSP-BF531 (84KB Total)	ADSP-BF532 (116KB Total)	ADSP-BF533 (148KB Total)
32KB Instruction ROM	32KB Instruction ROM	32KB Instruction SRAM
	32KB Instruction SRAM	32KB Instruction SRAM
16KB Instruction SRAM		
16KB Instr SRAM/Cache	16KB Instr SRAM/Cache	16KB Instr SRAM/Cache
		32KB Data SRAM
	16KB Data SRAM/Cache	16KB Data SRAM/Cache
16KB Data SRAM/Cache	16KB Data SRAM/Cache	16KB Data SRAM/Cache
4KB Scratchpad	4KB Scratchpad	4KB Scratchpad

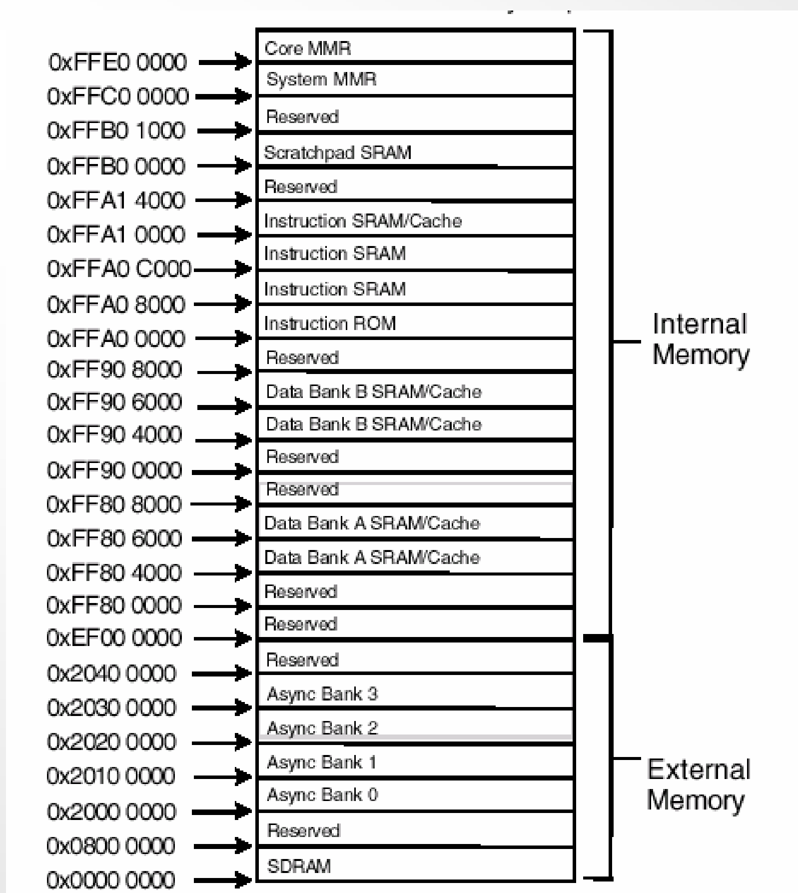
KAZTEK ENGINEERING

ANALOG DEVICES

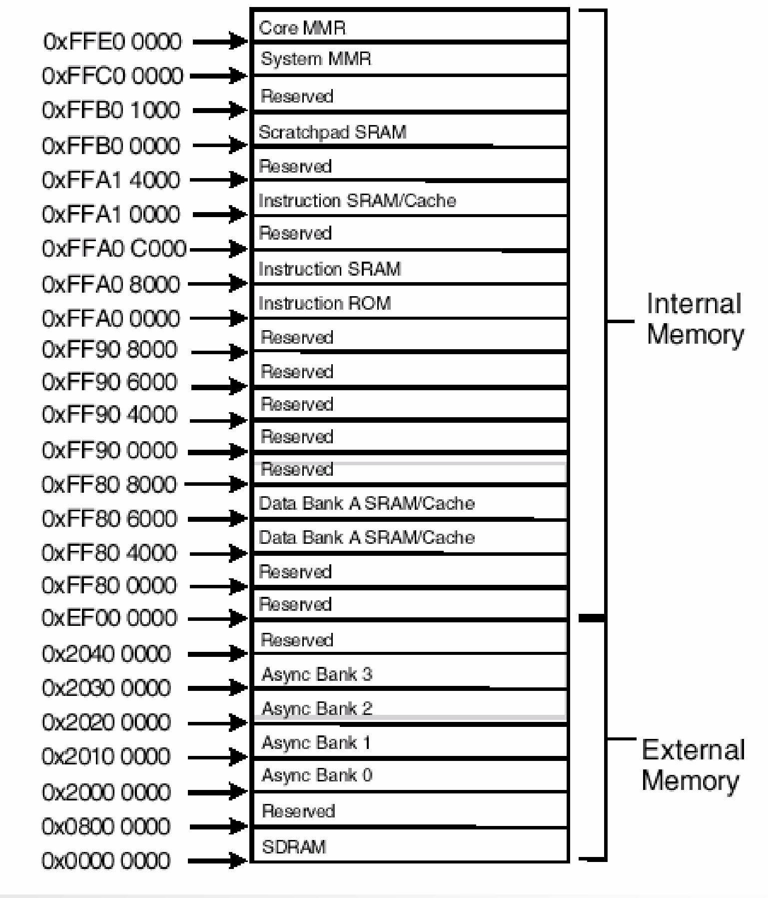
ADSP-BF533 Memory Map



ADSP-BF532 Memory Map

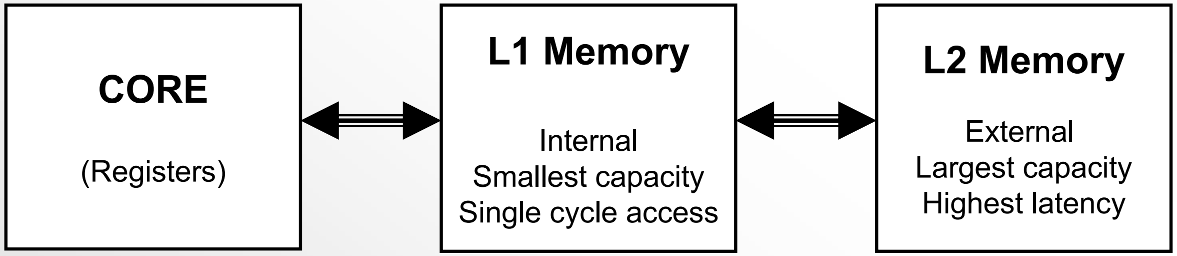


ADSP-BF531 Memory Map

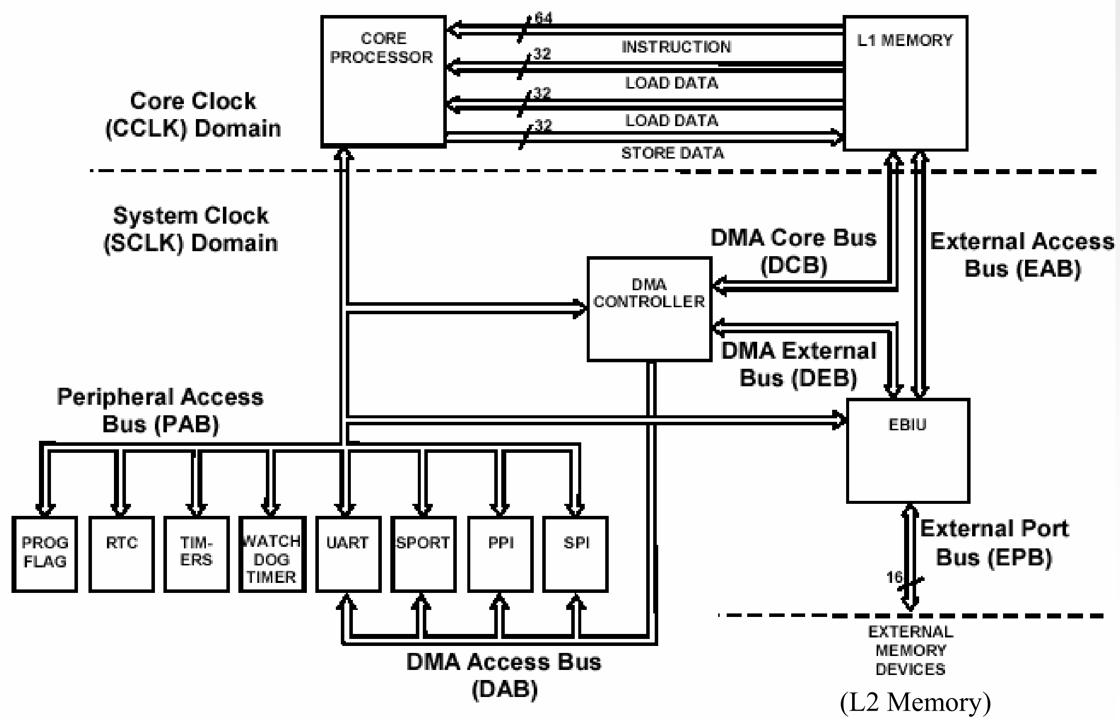


Memory Hierarchy on the BF533

- **As processor speeds increase (300Mhz – 1 GHz), it becomes increasingly difficult to have large memories running at full speed.**
- **The BF53x uses a *memory hierarchy* with a primary goal of achieving memory performance similar to that of the fastest memory (i.e. L1) with an overall cost close to that of the least expensive memory (i.e. L2)**



Internal Bus Structure of the ADSP-BF533



KAZTEK ENGINEERING

ANALOG DEVICES

Configurable Memory

- The best system performance can be achieved when executing code or fetching data out of L1 memory
- Two methods can be used to fill the L1 memory – Caching and Dynamic Downloading – Blackfin® Processor Supports Both.
  - Micro-controllers have typically used the caching method, as they have large programs often residing in external memory and determinism is not as important.
  - DSPs have typically used Dynamic Downloading as they need direct control over which code runs in the fastest memory.
- Blackfin® Processor allows the programmer to choose one or both methods to optimize system performance.

Why Do Blackfin® Processors Have Cache?

- **To allow users to take advantage of single cycle memory without having to specifically move instructions and or data “manually”**
  - **L2 memory can be used to hold large programs and data sets**
  - **The paths to and from L1 memory are optimized to perform with cache enabled**
- **Automatically optimizes code that reuses recently used or nearby data**

Internal L1 Memory: Smallest capacity Single cycle access	External L2 Memory: Largest capacity Highest latency

KAZTEK ENGINEERING

ANALOG DEVICES

Configurable L1 Memory Selections

L1 Instruction	L1 Data A	L1 Data B (BF533 and BF532 only)	L1 Data Scratchpad
Cache	Cache	Cache	SRAM
Cache	Cache	SRAM	SRAM
Cache	SRAM	SRAM	SRAM
SRAM	Cache	Cache	SRAM
SRAM	Cache	SRAM	SRAM
SRAM	SRAM	SRAM	SRAM

Using instruction cache will improve performance for most applications

Data Cache may or may not improve performance

Max bandwidth into L1 memory is available with cache enabled

Trade-offs must be made on code control and peak short-term performance

Core MMR L1 Memory Registers

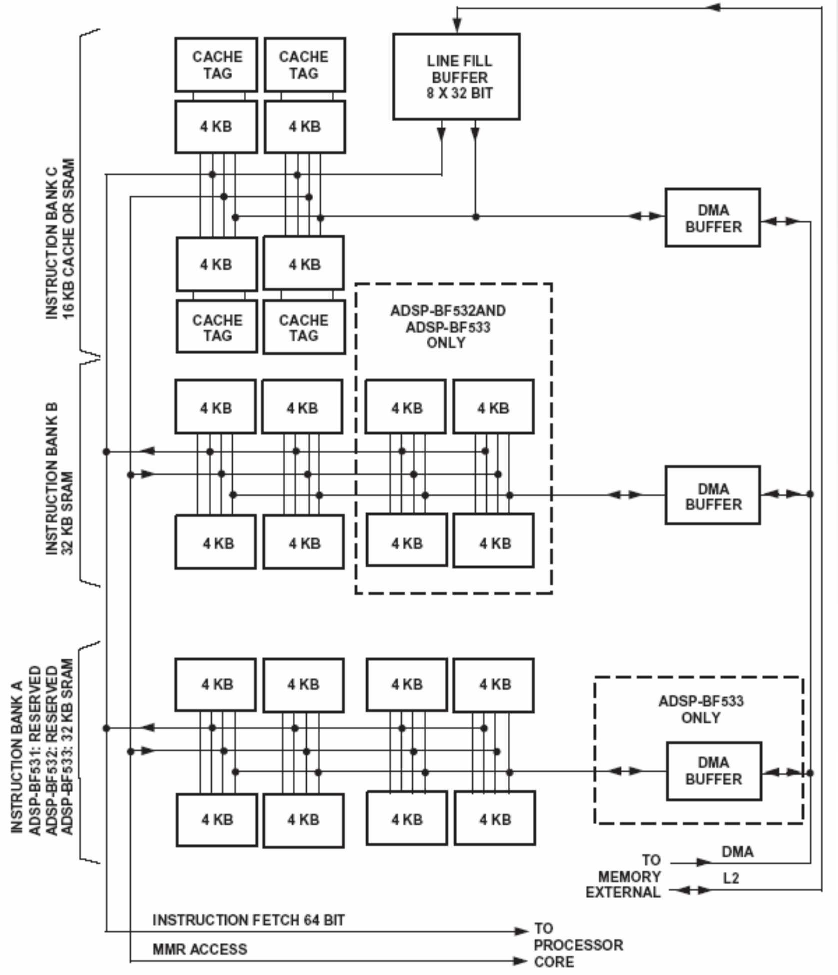
- General Control
  - IMEM_CONTROL (Instruction Memory)
  - DMEM_CONTROL (Data Memory)
- Cache and Protection Properties (n=0 to 15)
  - ICPLB_DATAn, ICPLB_ADDRn
  - DCPLB_DATAn, ICPLB_ADDRn
- Test Functionality
  - ITEST_COMMAND, ITEST_DATA
  - DTEST_COMMAND, DTEST_DATA

BF533 L1 Instruction Memory

Instruction Bank C
BF531, BF532, BF533:
16KB SRAM/CACHE

Instruction Bank B
BF531: 16KB SRAM
BF532: 32KB SRAM
BF533: 32KB SRAM

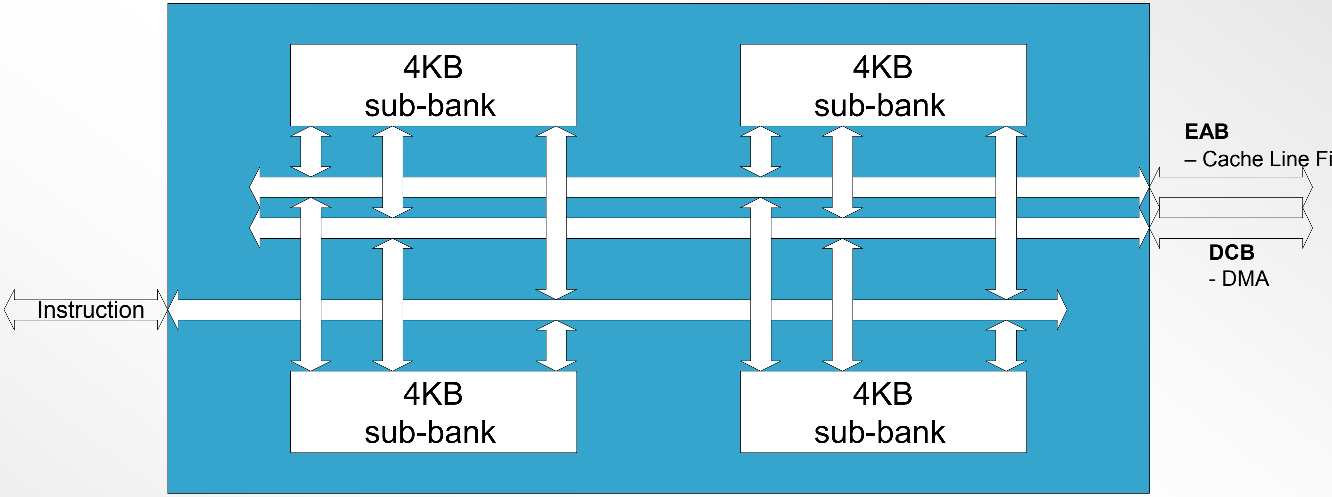
Instruction Bank A
BF531: 32KB ROM
BF532: 32KB ROM
BF533: 32KB SRAM



KAZTEK ENGINEERING

ANALOG DEVICES

L1 Instruction Memory 16KB Configurable Bank



16 KB SRAM

- Four 4KB single-ported sub-banks
- Allows simultaneous core and DMA accesses to different banks

16 KB cache

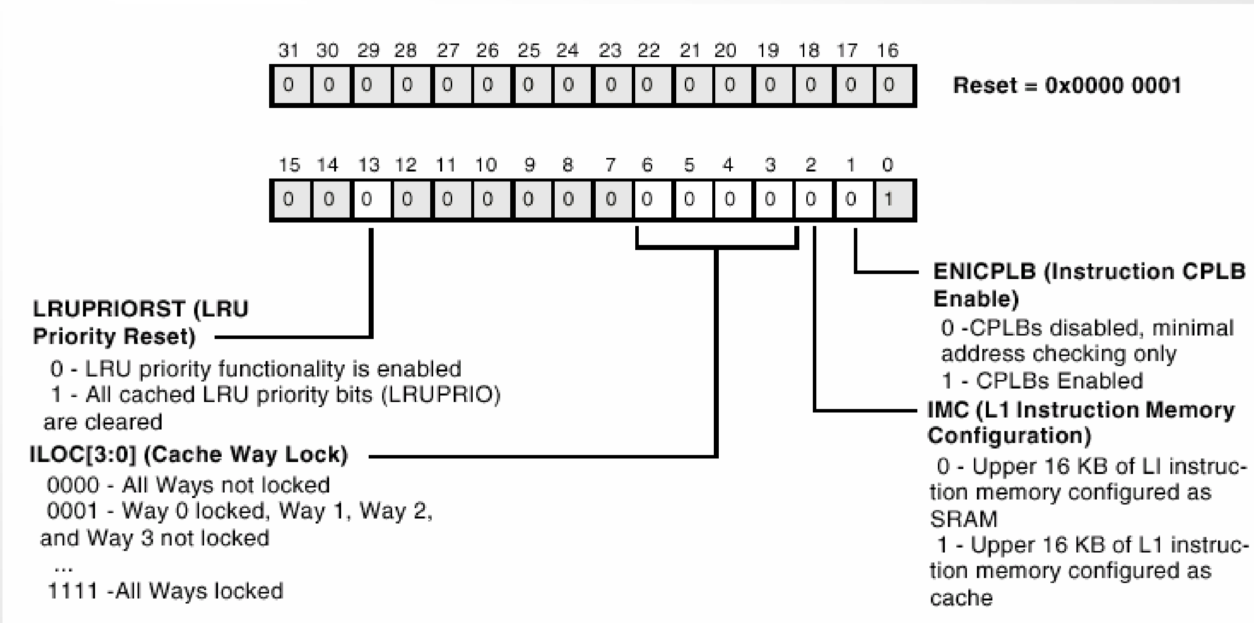
- 4-way set associative with arbitrary locking of ways and lines
- LRU replacement
- No DMA access

Features of L1 Instruction Memory Unit

- **Instruction Alignment Unit: handles alignment of 16-, 32-, and 64-bit instructions that are to be sent to the execution unit.**
- **Cacheability and Protection Look-aside Buffer (CPLB): Provides cacheability control and protection during instruction memory accesses.**
- **256-bit cache Line Fill Buffer: uses four 64-bit word burst transfers to copy cache lines from external memory.**
- **Memory test interface: Provides software with indirect access to tag and data memory arrays.**

L1 Instruction Memory Control Register

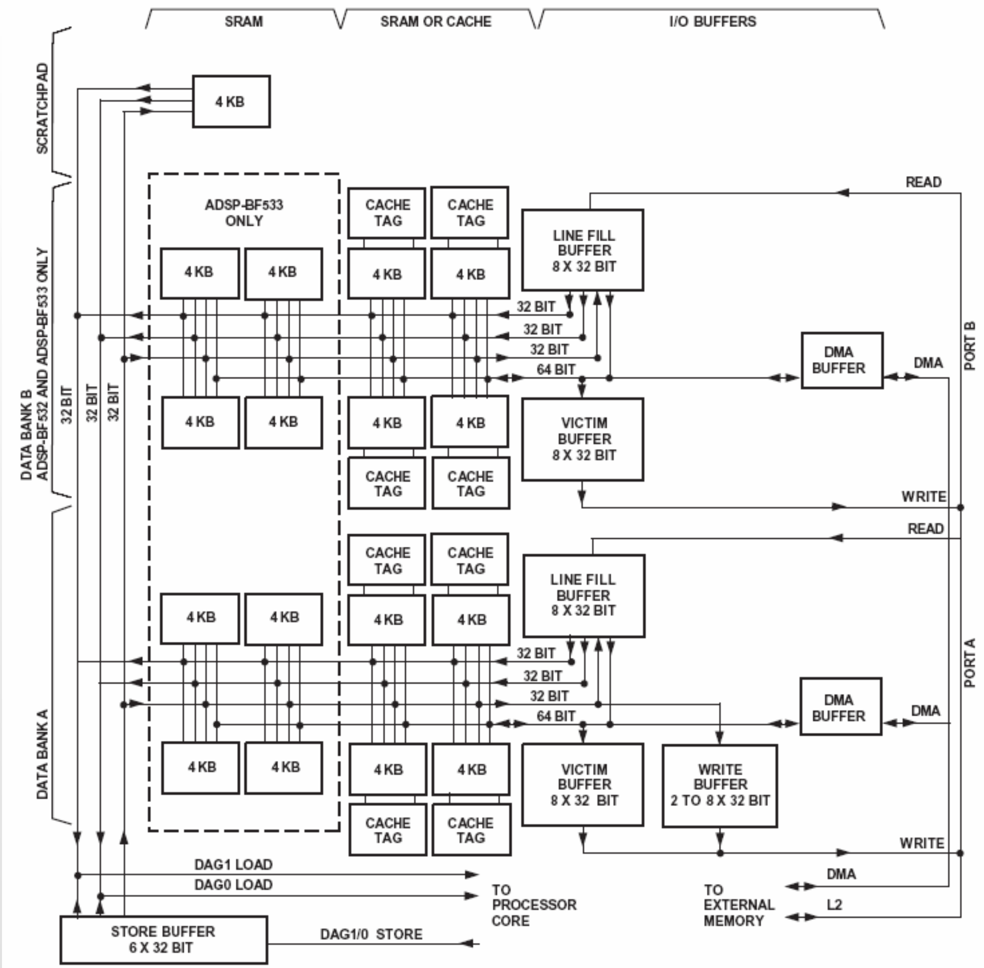
IMEM_CONTROL



KAZTEK ENGINEERING

ANALOG DEVICES

BF533 L1 Data Memory

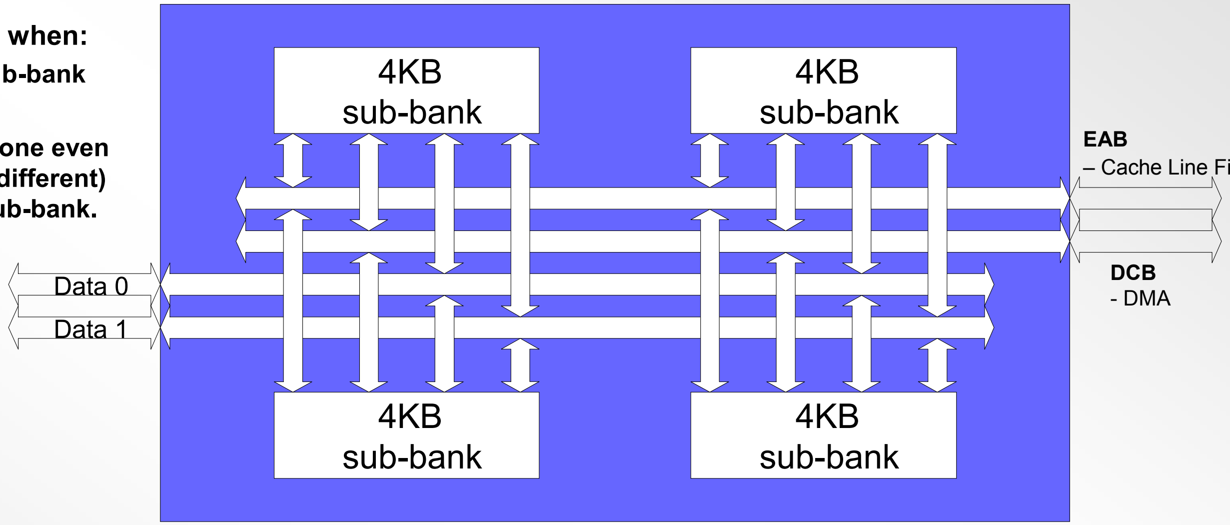


Victim Buffers:
Victimized Write-Back Cached Data to external memory

Write Buffer:
Write-Through and Non-cached Data to external memory

L1 Data Memory 16KB Configurable Bank

Block is Multi-ported when:
Accessing different sub-bank
OR
Accessing one odd and one even access (Addr bit 2 different) within the same sub-bank.



- When Used as SRAM
  - Allows simultaneous dual DAG and DMA access
- When Used as Cache
  - Each bank is 2-way set-associative
  - No DMA access
  - Allows simultaneous dual DAG access

BF533 L1 Data Memory

Sub-Bank	Data Bank A	Data Bank B	
1	0xFF80 0000	0xFF90 0000	SRAM
2	0xFF80 1000	0xFF90 1000	
3	0xFF80 2000	0xFF90 2000	
4	0xFF80 3000	0xFF90 3000	
5	0xFF80 4000	0xFF90 4000	CONFIGURABLE
6	0xFF80 5000	0xFF90 5000	
7	0xFF80 6000	0xFF90 6000	
8	0xFF80 7000	0xFF90 7000	

L1 configurable data memory can be:

- Both banks A & B as SRAM
- Bank A as cache, bank B as SRAM
- Both banks as cache

BF532 L1 Data Memory

Sub-Bank	Data Bank A	Data Bank B	
1	0xFF80 0000	0xFF90 0000	SRAM
2	0xFF80 1000	0xFF90 1000	
3	0xFF80 2000	0xFF90 2000	
4	0xFF80 3000	0xFF90 3000	
5	0xFF80 4000	0xFF90 4000	CONFIGURABLE
6	0xFF80 5000	0xFF90 5000	
7	0xFF80 6000	0xFF90 6000	
8	0xFF80 7000	0xFF90 7000	

L1 configurable data memory can be:

- Both banks A & B as SRAM
- Bank A as cache, bank B as SRAM
- Both banks as cache

BF531 L1 Data Memory

Sub-Bank	Data Bank A	Data Bank B	
1	0xFF80 0000	0xFF90 0000	SRAM
2	0xFF80 1000	0xFF90 1000	
3	0xFF80 2000	0xFF90 2000	
4	0xFF80 3000	0xFF90 3000	
5	0xFF80 4000	0xFF90 4000	CONFIGURABLE
6	0xFF80 5000	0xFF90 5000	
7	0xFF80 6000	0xFF90 6000	
8	0xFF80 7000	0xFF90 7000	

L1 configurable data memory can be:

- •Bank A as SRAM
- •Bank A as Cache

L1 Data Memory SRAM Addressing

- Both DAG units can access Data Banks A & B
- If an address conflict is detected Data Bank priority is as follows:
  1. System DMA (highest priority)
  2. DAG Unit 0
  3. DAG Unit 1 (lowest priority)
- Parallel DAG accesses can occur to the same Data Bank as long as the references are to different sub-banks OR they access 2 words of different 32-bit address polarity (Address bit 2 is different).

Dual Access to Same Sub-Bank

A2 = 1 (odd)				A2 = 0 (even)			
07	06	05	04	03	02	01	00
0F	0E	0D	0C	0B	0A	09	08
17	16	15	14	13	12	11	10
1F	1E	1D	1C	1B	1A	19	18
27	26	25	24	23	22	21	20
2F	2E	2D	2C	2B	2A	29	28

A dual access to an odd and even (quad address) location can be performed in a single cycle

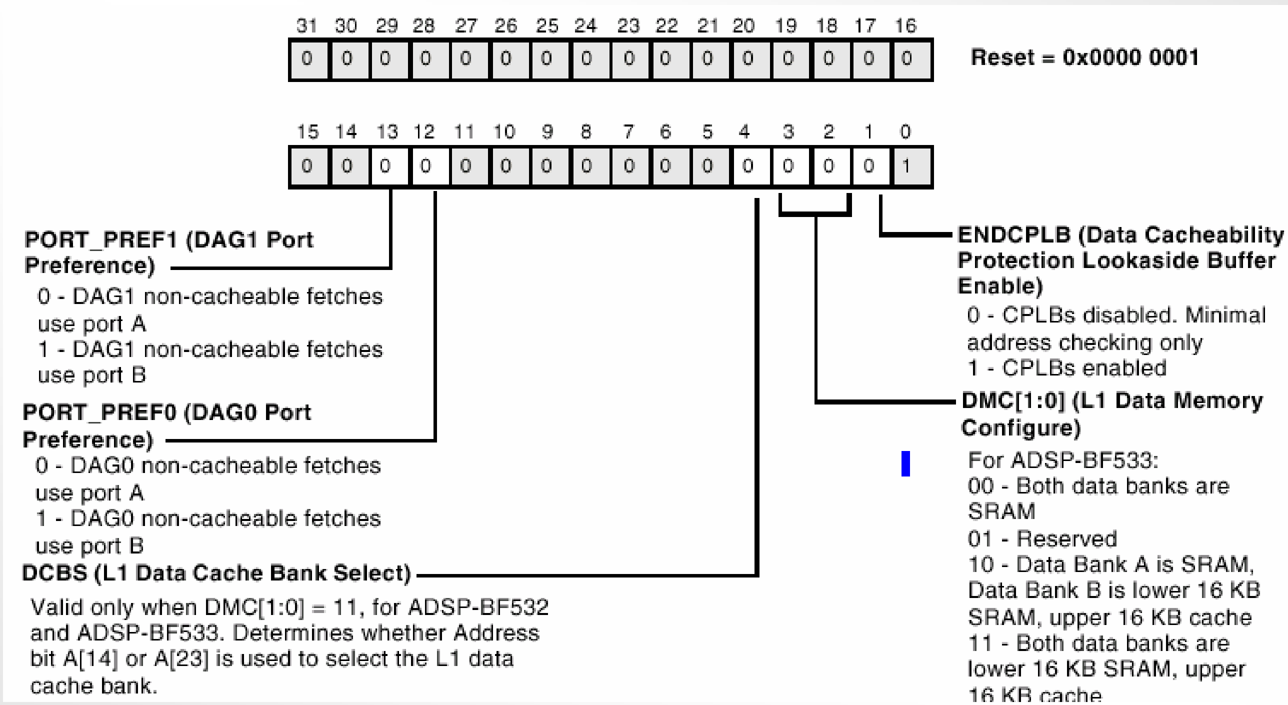
A dual access to two odd or two even locations will result in an extra cycle (1 stall) of delay

L1 Scratchpad Memory

- Dedicated 4KB Block of Data SRAM
- Operates at CCLK rate
- Can not be configured as Cache
- Can not be accessed by DMA
- Typical Use is for User and Supervisor stacks to do fast context switching during interrupt handling.

L1 Data Memory Control Register

DMEM_CONTROL



Cache Mode

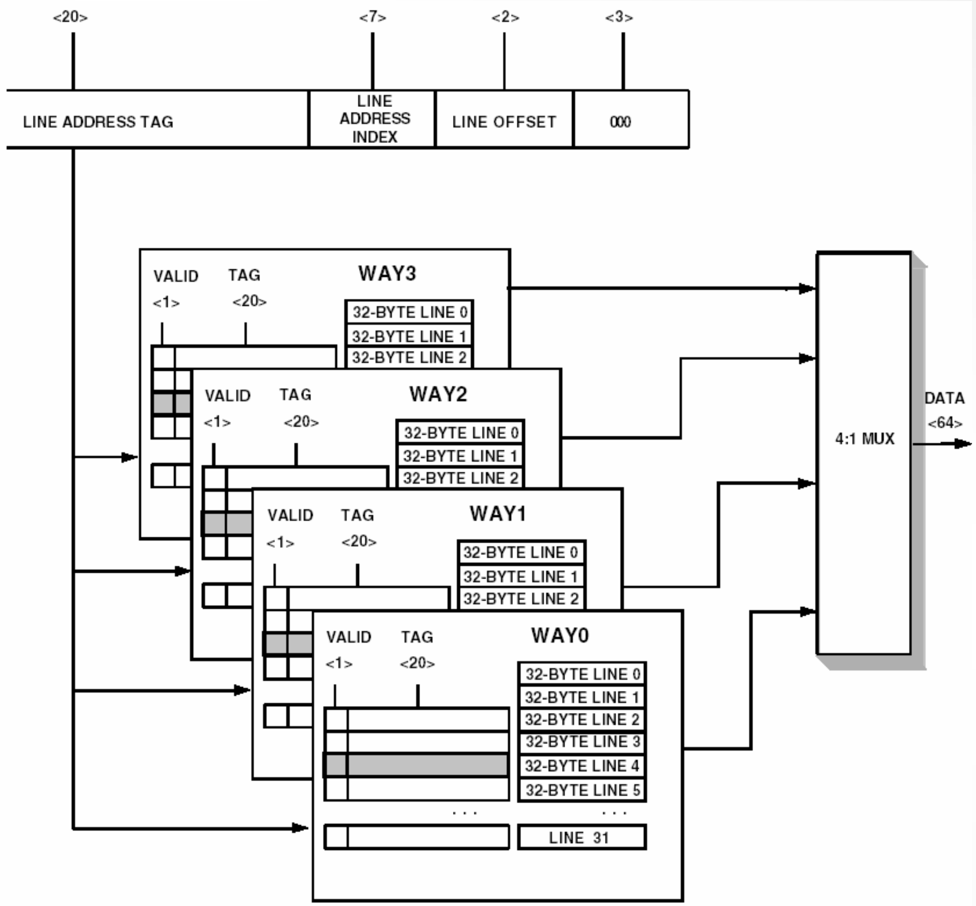
What is Cache?

- **In a hierarchical memory system, *cache* is the first level of memory reached once the address leaves the core (i.e L1)**
  - **If the instruction/data word (8, 16, 32, or 64 bits) that corresponds to the address is in the cache, there is a *cache hit* and the word is forwarded to the core from the cache.**
  - **If the word that corresponds to the address is not in the cache, there is a *cache miss*. This causes a fetch of a fixed size block (which contains the requested word) from the main memory.**
    - **The Blackfin allows the user to specify which regions (i.e. *pages*) of main memory are cacheable and which are not through the use of CPLBs (more on this later).**
      - **If a page is cacheable, the block (i.e. *cache line* containing 32 bytes) is stored in the cache after the requested word is forwarded to the core**
      - **If a page is non-cacheable, the requested word is simply forwarded to the core**

KAZTEK ENGINEERING

ANALOG DEVICES

ADSP-BF533 Instruction Cache Organization

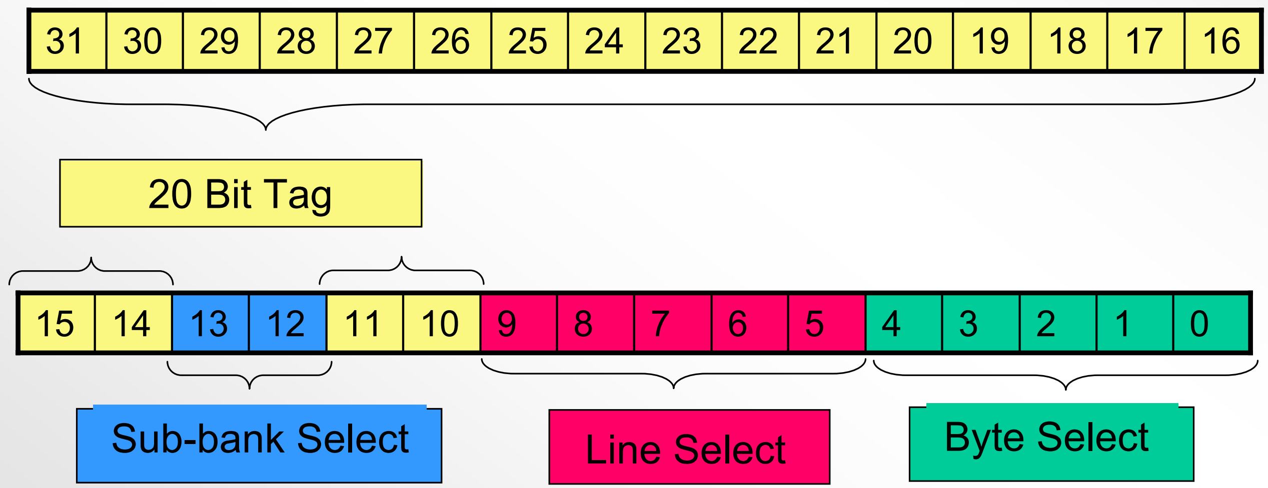


SHADED BOXES ACROSS EACH WAY CONSTITUTE A SET.

- **Cache Line:**
  - A 32 byte contiguous block of memory
- **Set:**
  - A group of cache lines in the cache
    - Selected by Line Address Index
- **Way:**
  - One of several places in a set that a cache line can be stored
    - 1 of 4 for Instructions
    - 1 of 2 for Data
- **Cache Tag:**
  - Upper address bits stored with cache line. Used to ID the specific address in main memory that the cached line represents

Instruction Cache Placement Based On Address

- Four 4KB sub-banks (16KB total)
- Each sub-bank has 4-ways (1KB for each way)
- Each way has 32 lines
- Each line is 32 bytes



Cache Hits and Misses

- A cache hit occurs when the address for an instruction fetch request from the core matches a valid entry in the cache.
- A cache hit is determined by comparing the upper 18 bits, and bits 11 and 10 of the instruction fetch address to the address tags of valid lines currently stored in a cache set.
- Only valid cache lines (i.e. cache lines with their valid bits set) are included in the address tag compare operation.
- When a cache hit occurs, the target 64-bit instruction word is sent to the instruction alignment unit where it is stored in one of two 64-bit instruction buffers.
- When a cache miss occurs, the instruction memory unit generates a cache line-fill access to retrieve the missing cache line from external memory to the core.

Instruction Fill from L2 Memory

- Cache Off
  - 64 bits

64 bits

- Cache On
  - Burst Cache Line fill (32-bytes)

64 bits	64 bits	64 bits	64 bits

KAZTEK ENGINEERING

ANALOG DEVICES

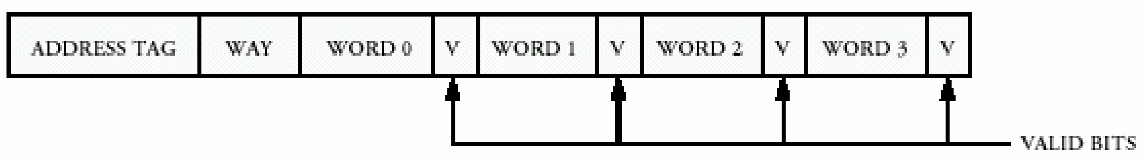
Cache Line Fills

- A cache line fill consists of fetching 32 bytes of data from memory external to the core (i.e. L2 memory).
- A line read data transfer consists of a four 64-bit word read burst.
- The instruction memory unit requests the target instruction word first; once it has returned the target word the IMU requests the next three words in sequential address order and wrap around if necessary.

Target Word	Fetching Order for Next Three Words
WD0	WD0, WD1, WD2, WD3
WD1	WD1, WD2, WD3, WD0
WD2	WD2, WD3, WD0, WD1
WD3	WD3, WD0, WD1, WD2

Cache Line-Fill Buffer

- The cache line-fill buffer allows the core to access the data from the new cache line as the line is being retrieved from external memory, rather than having to wait until the line has been completely written to the 4KB memory block.
- The line-fill buffer organization is shown below:



- The line-fill buffer is also used to support non-cacheable accesses*. A non-cacheable access consists of a single 64-bit transfer on the instruction memory unit's external read port.

* A non-cacheable access includes: external accesses when instruction memory is configured as SRAM, or accesses to non-cacheable pages

Cache Line Replacement

- **The cache line replacement unit first checks for invalid entries.**
- **If only a single invalid entry is found then that entry is selected for the new cache line. If multiple invalid entries are found the replacement entry for the new cache line is selected based on the following priority:**
  - **way 0 first**
  - **way 1 next**
  - **way 2 next**
  - **way 3 last**
- **When no invalid entries are found, the cache replacement logic uses a 6-bit LRU algorithm to select the entry for the new cache line.**
- **For instruction cache the LRUPRIO bit is also considered.**

Instruction Cache “Locking By Line” (LRUPRIO)

- Prevents the Cached Line from being replaced
- CPLB_LRUPRIO bits in the ICPLB_DATAx register define the priority for that page.
- The Cache line importance level (LRUPRIO) is saved in the TAG and used by the replacement policy logic.
- Cache Line Replacement policy with LRUPRIO
  - No invalid entries:
    - A high priority will replace a low priority or a high priority if all 4-ways contain high priority lines.
    - LRU (least recently used) policy is used to determine which one of the lines that have the same priority will be replaced.
- Setting the IMEM_CONTROL: LRUPRIORST bit clears all LRUPRIO bits in the TAGs.

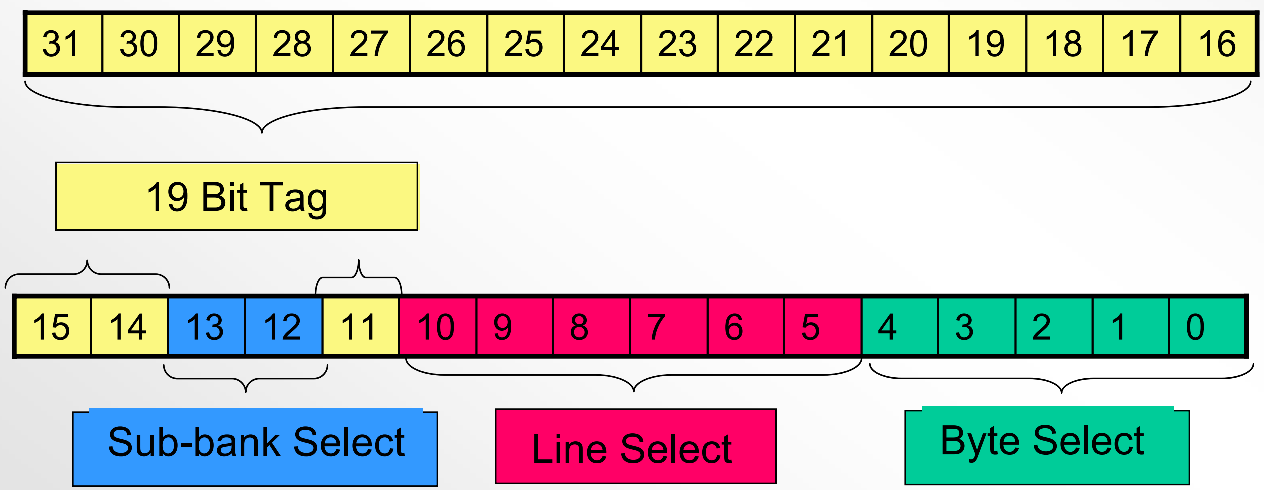
Instruction Cache Locking By Way

- Each 4KB way of the instruction cache can be locked individually to ensure placement of performance-critical code.
- Controlled by the ILOC<3:0> bits in the IMEM_CONTROL register.

Data Cache Mode

Data Cache Placement Based On Address

- Four 4KB sub-banks (16KB total)
- Each sub-bank has 2-ways (2KB for each way)
- Each way has 64 lines
- Each line is 32 bytes
- If Both Data Bank A and B are set for Cache, bit 14 or 23 is used to determine which Data Bank.



KAZTEK ENGINEERING

ANALOG DEVICES

Data Cache Definitions

- **Write Through:**
  - A cache write policy where write data is written to the cache line and to the source memory.
- **Write Back:**
  - A cache write policy where write data is written only to the cache line. The modified cache line is written to source memory only when it is replaced.
- **Dirty/Clean (Applies to Write Back Mode only):**
  - State of cache line indicating whether the data in the cache has changed since it was copied from source memory

- **Performance trade-off required between write through and write back to determine the best policy to use for an application.**

Data Cache Victim Buffer

- **The victim buffer is used to read a dirty cache line either being flushed or replaced by a cache line fill and then to initiate a burst write operation on the bus to perform the line copyback to the system.**
- **The processor can continue running without having to wait for the data to be written back to L2 memory.**
- **The victim buffer is comprised of a 4-deep FIFO each 64-bits in width (similar to the fill-buffer.)**
- **There is no data forwarding support from the victim buffer.**

Cacheability Protection Lookaside Buffers (CPLBS)

Memory Protection and Cache Properties

- Memory Management Unit
  - Cacheability and Protection Look-Aside Buffers (CPLBs)
  - Cache/protection properties determined on a per memory page basis (1K, 4K, 1M, 4M byte sizes)
  - 32 CPLBs total: 16 CPLBs for instruction memory, 16 CPLBs for data memory
- User/Supervisor Access Protection
- Read/Write Access Protection
- Cacheable or Non-Cacheable

Using CPLBs

- **Cache enabled:**
  - CPLB <u>must</u> be used to define cacheability properties
- **Cache disabled:**
  - CPLBs <u>can</u> be used to protect pages of memory

• When CPLBS are enabled, a valid CPLB must exist before an access to a specific memory location is attempted. Otherwise, an exception will be generated.

• User and Supervisor mode protection is available without using CPLBs.

KAZTEK ENGINEERING

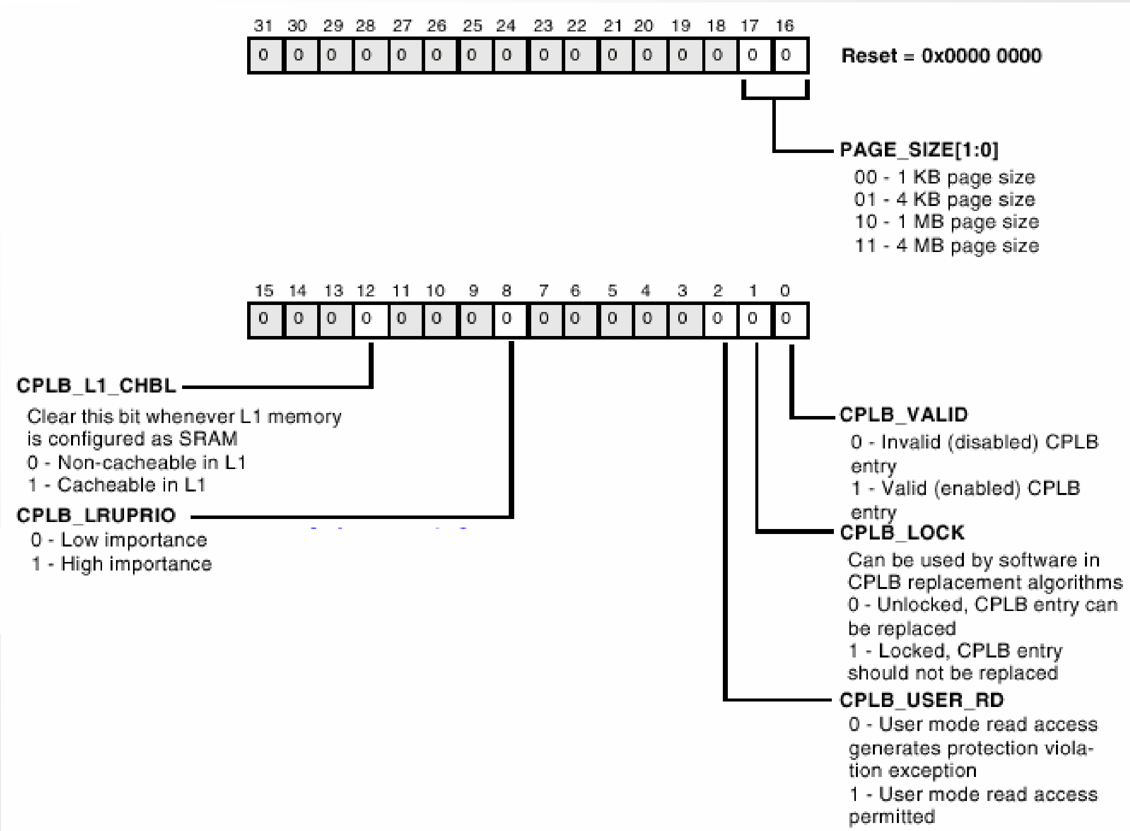
ANALOG DEVICES

Cacheability Protection Lookaside Buffers (CPLBs)

- Divide the entire Blackfin memory map into regions (i.e. pages) that have cacheability and protection properties.
- 16 Pages in Instruction Memory plus 16 Pages in Data memory
  - Page sizes: 1KB, 4KB, 1MB, 4MB
- Each CPLB has 2 associated registers:
  - 32bit Start Address: ICPLB_ADDRn, DCPLB_ADDRn
  - Cache/Protection Properties: ICPLB_DATAn, DCPLB_DATAn

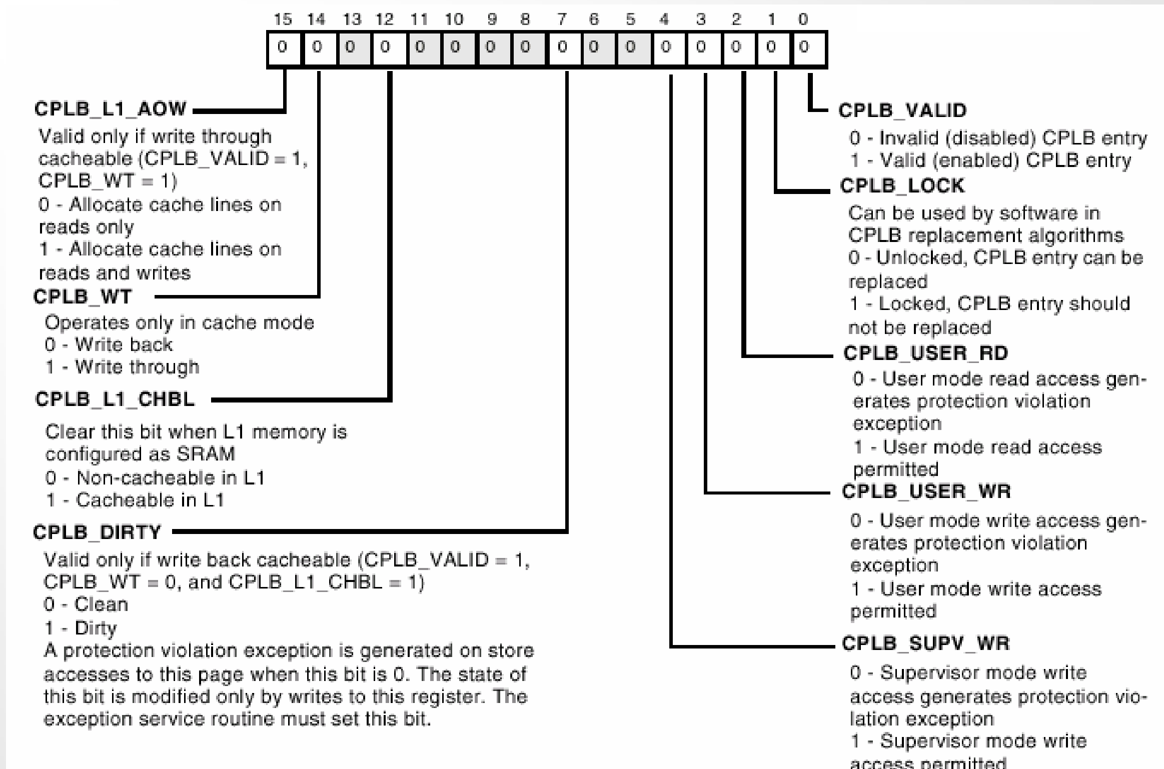
Note: “n” equals 15:0

ICPLB_DATAn Register



Note: "n" equals 15:0

DCPLB_Datan Register



*Bits 17:16 Page Size[1:0] same as ICPLB Register

Note: “n” equals 15:0

Example Protection Operation

- Set up CPLBs to define regions and properties:
  - Default hardware CPLBs are present for MMRs and scratchpad memory.
  - CPLBs must be configured for L1 Data and L1 Instruction Memory as Non-Cacheable
  - Disable all memory other than the desired memory space.
  - Execute Code.
- If code tries to access memory that has been ‘disabled’ or protected, then a ‘memory protection violation’ occurs as an exception.

Example CPLB Setup

Instruction CPLB setup

L1 Instruction: Non-cacheable
1MB page

SDRAM: Cacheable
Eight 4MB pages

Async: Non-cacheable
One 4MB page

Async: Cacheable
Two 4MB pages

Memory management handles exceptions and redefines external memory pages as required for external memory. Examples will be provided to customers.

Data CPLB setup

L1 Data: Non-cacheable
One 4MB page

SDRAM: Cacheable
Eight 4MB pages

Async: Non-cacheable
One 4 MB page

Async: Cacheable
One 4 MB page

KAZTEK ENGINEERING

ANALOG DEVICES

Accessing the Cache Directly

- Once L1 memory is configured as cache, it can’t be accessed via DMA or from a core read.
- ITEST_COMMAND and ITEST_DATA memory mapped registers do allow direct access to Instruction Memory tags and lines.
- Analogous registers exist for Data Cache.
- Can be useful for invalidating cache lines directly.

Data Cache Control Instructions

- **Prefetch: Causes data cache to prefetch line associated with address in P-register**
  - **Causes line to be fetched if it is not currently in the cache and the location is cacheable**
  - **Otherwise it behaves like a nop**
    - **Prefetch [p2];**
    - **Prefetch [p2 ++]; // post increment by cache-line size**
- **FLUSH: Causes data cache to synchronize specified cache line with higher levels of memory**
  - **If the line is dirty, it is written out and marked clean**
    - **flush [p2];**
    - **flush [p2 ++]; // post increment by cache-line size**
- **FLUSHINV: Causes data cache to invalidate a specific line in cache.**
  - **If the line is dirty, it is written out:**
    - **flushinv [p2];**
    - **flushinv [p2 ++]; // post increment by cache-line size**

KAZTEK ENGINEERING

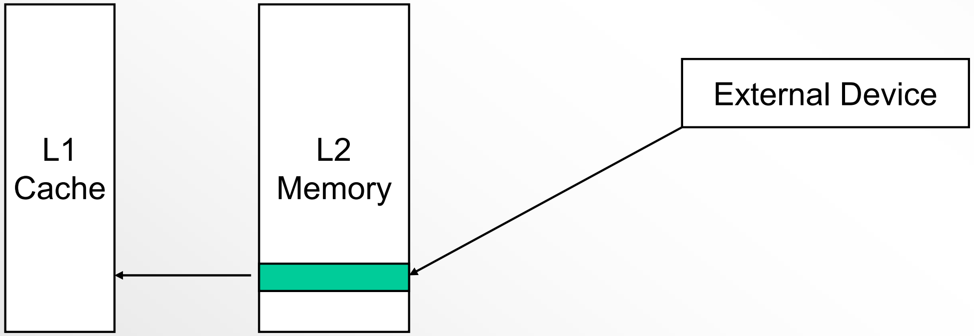
ANALOG DEVICES

Instruction Cache Control Instructions

- **IFLUSH: Causes instruction cache to invalidate a specific line in cache.**
  - **iflush [p2];**
  - **iflush [p2 ++]; // post increment by cache-line size**

Coherency Considerations

- Care must be taken when memory that is defined as “cacheable” is modified by outside source
  - DMA controller (data or descriptors)
- Cache is not aware of these changes so some mechanism must be setup
  - Simple memory polling will not work
  - Must Invalidate the cache before accessing the changed L2 memory.



Reference Material

Memory

Data Byte-Ordering

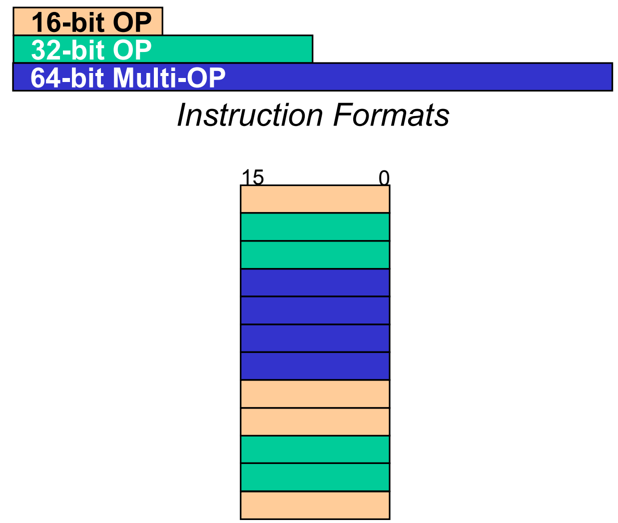
- The ADSP-BF533 architecture supports little-endian byte-ordering
- For example, if the hex value 0x76543210 resides in register r0 and the pointer register p0 contains address 0x00ff0000, then the instruction "[p0] = r0;" would cause the data to be written to memory as shown below:

Byte Address	Data
0x00ff0000	0x10
0x00ff0001	0x32
0x00ff0002	0x54
0x00ff0003	0x76

- When loading a byte, half-word, or word from memory to a register, the LSB (bit 0) of the data word is always loaded into the LSB of the destination register

Instruction Packing

- **Instruction set tuned for compact code:**
  - **Multi-length instructions**
    - **16, 32, 64-bit opcodes**
    - **Limited multi-issue instructions**
- **No memory alignment restrictions for code:**
  - **Transparent alignment H/W.**



16-bit wide memory

No Memory Alignment Restrictions: Maximum Code Density and Minimum System Memory Cost

KAZTEK ENGINEERING

ANALOG DEVICES

Instruction Fetching

- 64-bit instruction line can fetch between 1 and 4 instructions

<table>
<tr><td colspan="4">One 64-bit instruction</td></tr>
<tr><td colspan="2">One 32-bit instruction</td><td colspan="2">One 32-bit instruction</td></tr>
<tr><td>One 16-bit instruction</td><td>One 16-bit instruction</td><td>One 16-bit instruction</td><td>One 16-bit instruction</td></tr>
<tr><td colspan="2">One 32-bit instruction</td><td>One 16-bit instruction</td><td>One 16-bit instruction</td></tr>
</table>