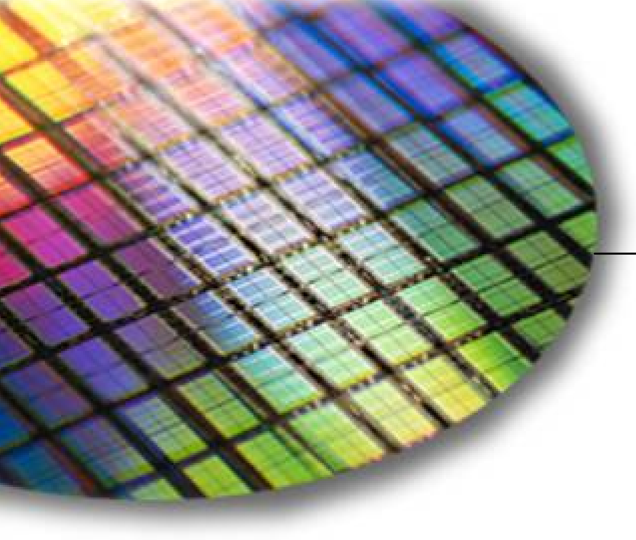


The World Leader in High Performance Signal Processing Solutions

Core Architecture Overview

Presented by:

George Kadziolka

President

Kaztek Systems

About This Module

This module introduces the Blackfin® family and provides an overview of the Blackfin processor architecture.

Module Outline

Blackfin Family Overview

The Blackfin Core

- **Arithmetic operations**
- **Data fetching**
- **Sequencing**

The Blackfin Bus Architecture and Memory

- **Modified Harvard architecture**
- **Hierarchical memory structure**
- **Flexible memory management**

Additional Blackfin Core Features

- **DMA**
- **Dynamic power management**
- **On-chip debug support**

Summary

Blackfin Family Overview

The Blackfin family consists of:

- A broad range of Blackfin processors
- Software development tools
- Hardware evaluation and debug tools

Extensive third-party support

- Development tools
- Operating systems
- TCP/IP stacks
- Hardware building blocks
- Software solutions

Blackfin Processors

All Blackfin processors combine extensive DSP capability with high end MCU functions on the same core.

- **Creates a highly efficient and cost-effective solution.**
- **A single software development tool chain**

All Blackfin processors are based on the same core architecture.

- **Once you understand one Blackfin processor, you can easily migrate from one family member to another.**
- **Code compatible across family members.**

Processors vary in clock speed, amount of on-chip memory, peripheral suite, package types and sizes, power, and price.

- **Large selection lets you optimize your choice of a Blackfin processor for your application.**

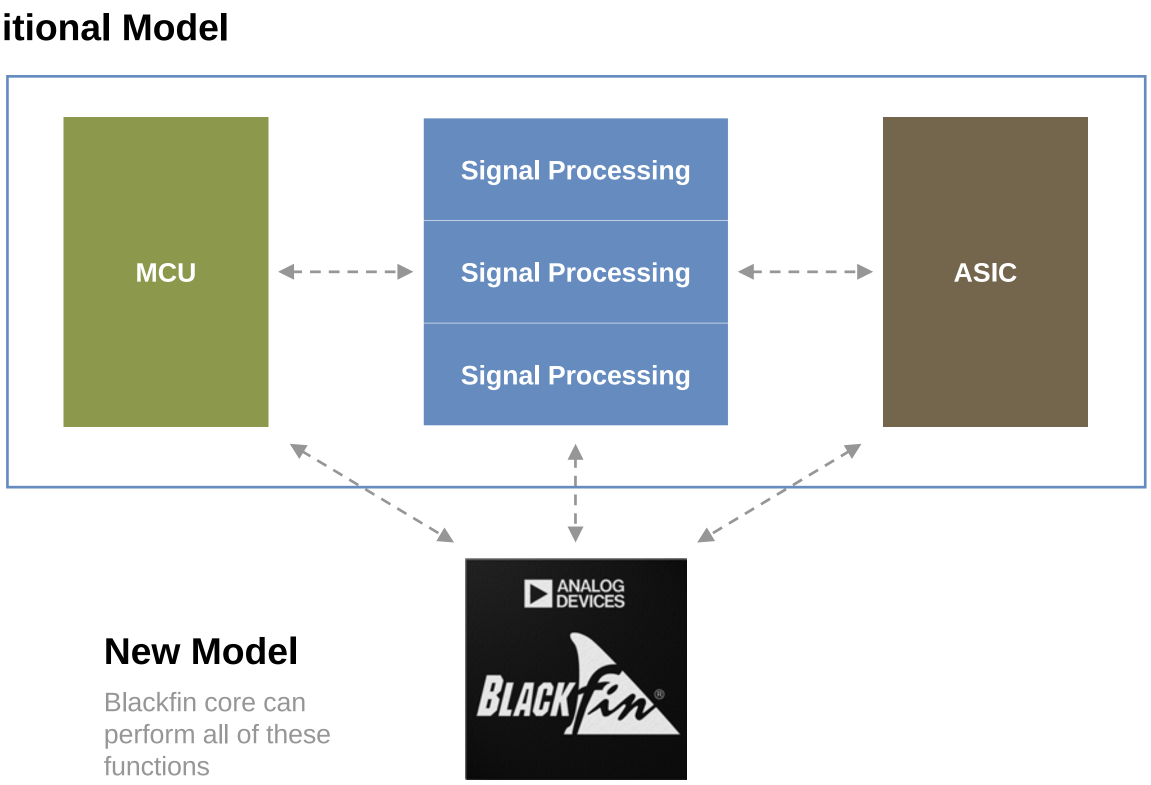
Blackfin Family Peripherals

The Blackfin family supports a wide variety of I/O:

- **EBIU (External Bus Interface Unit)**
- **Parallel peripheral interface (PPI)**
- **Serial ports (SPORTS)**
- **GPIO**
- **Timers**
- **UARTS**
- **SPI®**
- **Ethernet**
- **USB**
- **CAN®**
- **Two Wire Interface (TWI)**
- **Pixel compositor**
- **Lockbox™ secure technology**
- **Host DMA**
- **ATAPI**
- **SDIO**

See the Blackfin selection guide for complete details

Blackfin Processors Perform Signal Processing and Microcontroller Functions



MCU

- Control
- Networking
- RTC
- Watchdog
- RTOS
- MMU
- Byte addressable

ASIC

- Interfaces to sensors
- Broad peripheral mix
- Memory

New Model

Blackfin core can perform all of these functions

Blackfin Architecture
What does it mean for the developer?

Combining controller and DSP capabilities into a single core, along with rich I/O, enables development of efficient, low cost embedded media applications.

- **For example, multimedia over IP, digital cameras, telematics, software radio**

From a development perspective, a single core means there is only one tool chain.

- **An embedded application consisting of both control and signal processing modules is built using the same compiler.**
- **The result is dense control code and high performance DSP code.**

Features

Controller

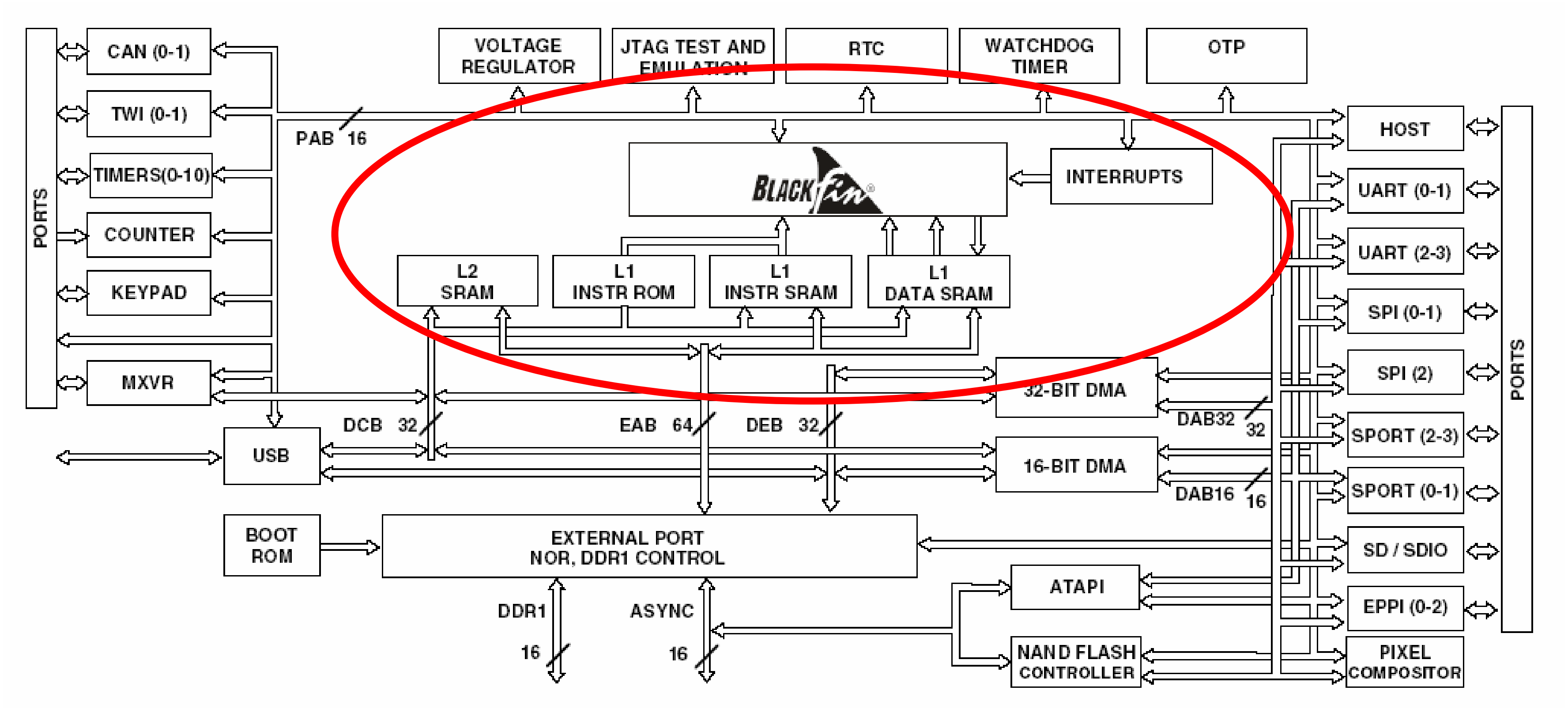
- L1 memory space for stack and heap
- Dedicated stack and frame pointers
- Byte addressability
- Simple bit-level manipulation

DSP

- Fast, flexible arithmetic computational units
- Unconstrained data flow to/from computational units
- Extended precision and dynamic range
- Efficient sequencing
- Efficient I/O processing
- The DSP aspect of the Blackfin core is optimized to perform FFTs and convolutions

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

Blackfin Core (e.g., ADSP-BF54x)



The Blackfin Core

The Blackfin Core

The core consists of:

Arithmetic unit

- Supports SIMD operation
- Load/store architecture

Addressing unit

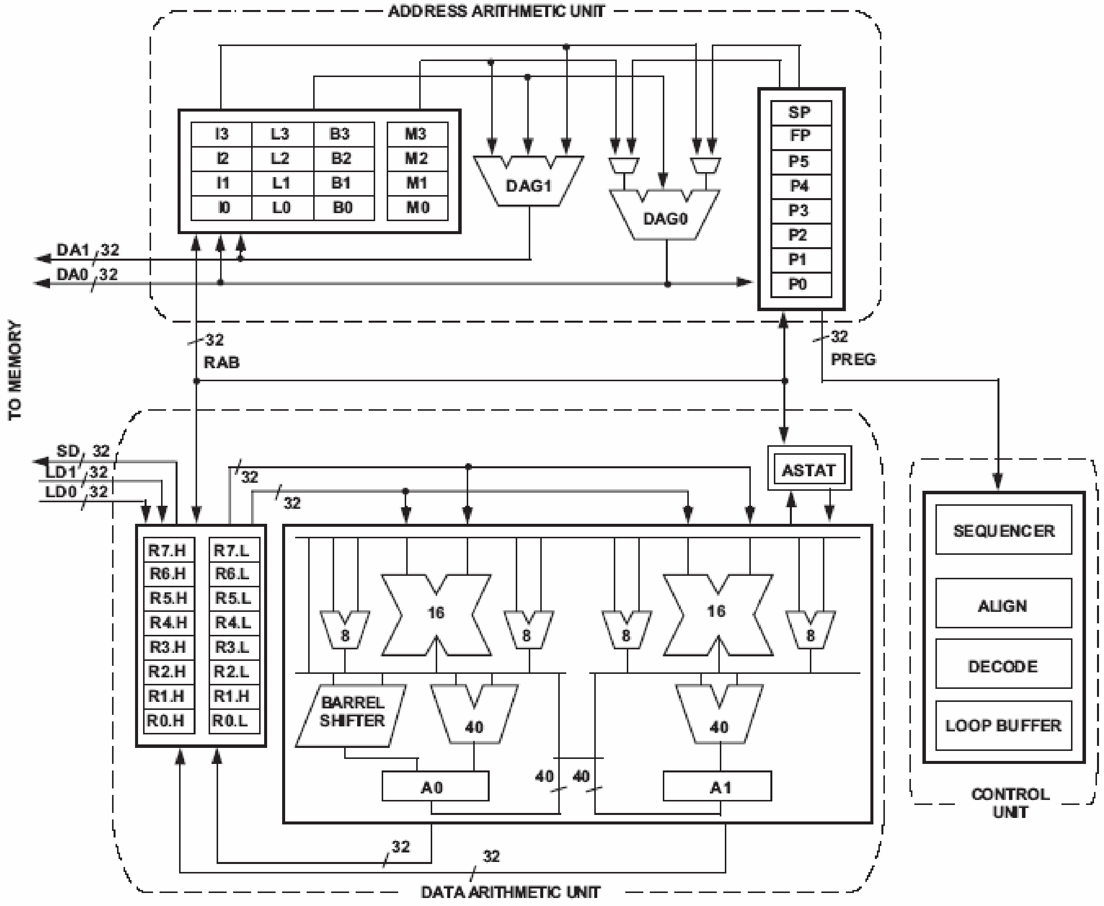
- Supports dual data fetch

Sequencer

- Efficient program flow control

Register files

- Data
- Addressing



The Arithmetic Unit

Performs arithmetic operations

Dual 40-bit ALU (Arithmetic/ Logic Unit)

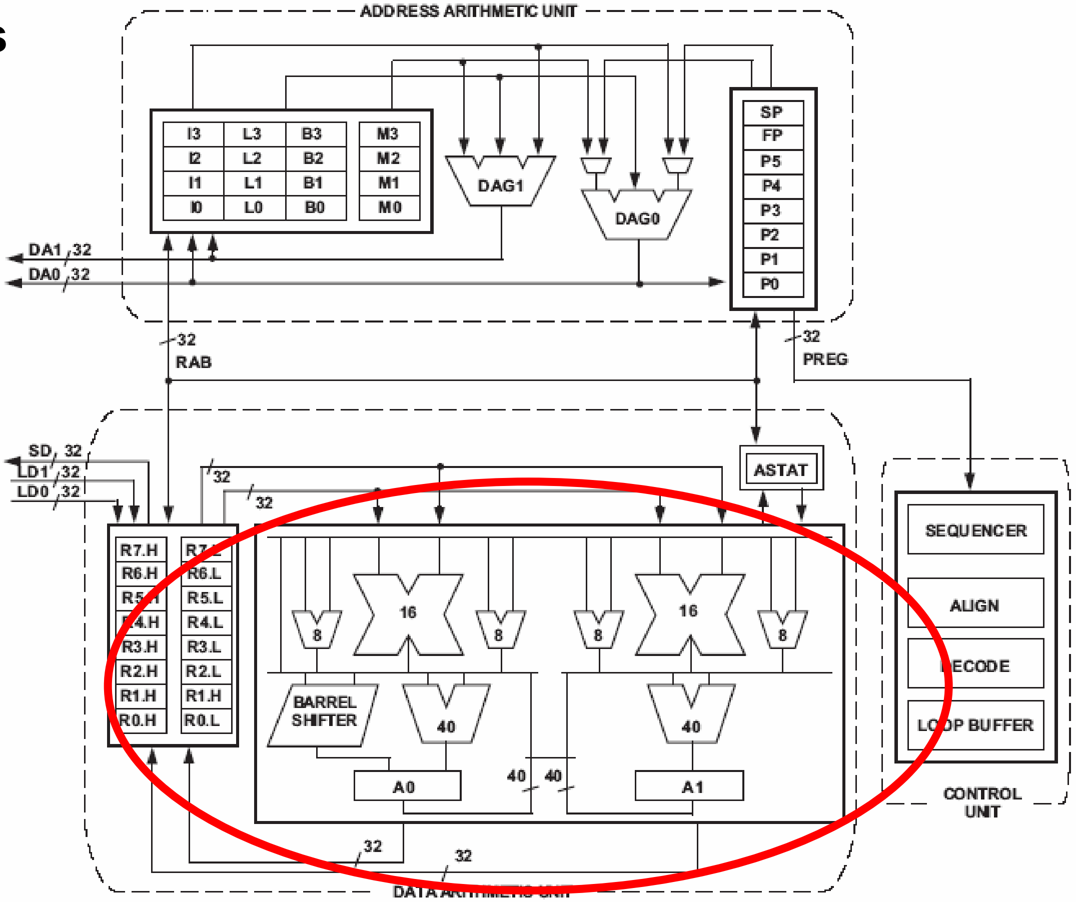
- Performs 16-/32-/40-bit arithmetic and logical operations

Dual 16 x 16 multiplier

- Performs dual MACs (multiply-accumulates) when used with ALUs

Barrel shifter

- Performs shifts, rotates, bit operations



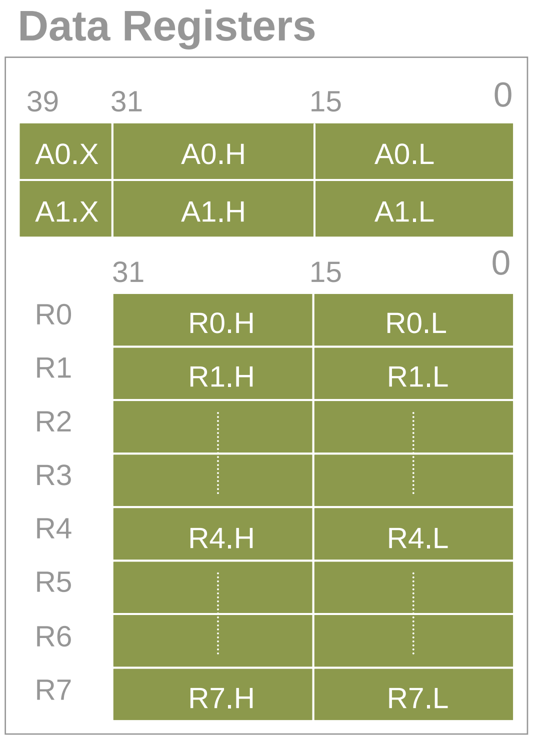
Data Registers

There are 8x 32-bit registers in the data register file.

- **Used to hold 32-bit vales or packed 16-bit**

There are also 2x 40-bit accumulators.

- **Typically used for MAC operations**

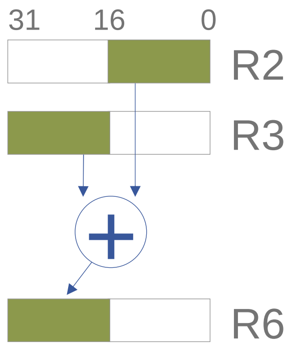


16-Bit ALU Operations—Examples

The Algebraic Assembly syntax is intuitive and makes it easy to understand what the instruction is doing.

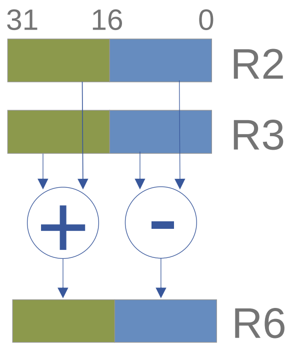
Single 16-bit operation

R6.H = R3.H + R2.L (s);



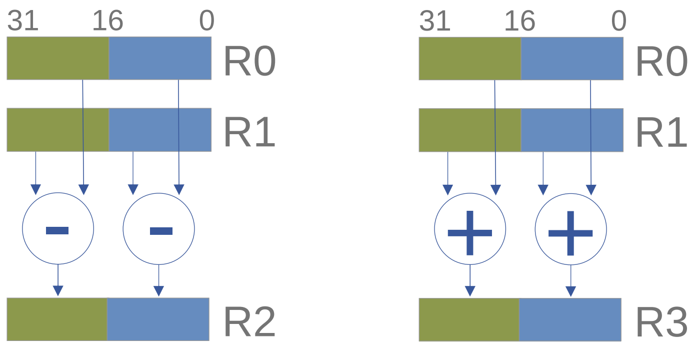
Dual 16-bit operation

R6 = R2 + | - R3;



Quad 16-bit operation

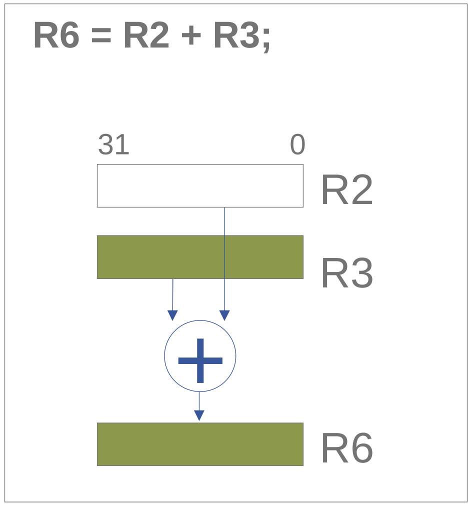
R3 = R0 - | - R1, R2 = R0 + | + R1;



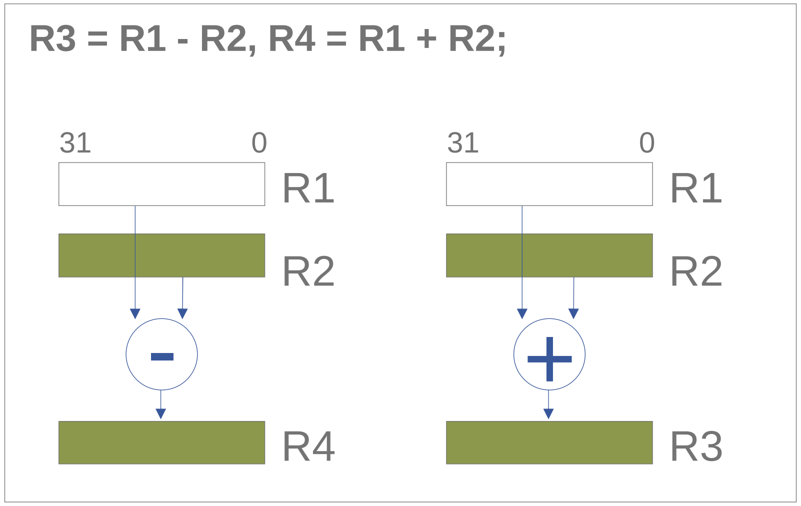
These operations effectively execute in a single cycle.

32-Bit ALU Operations—Examples

Single 32-bit addition



Dual 32-bit operation

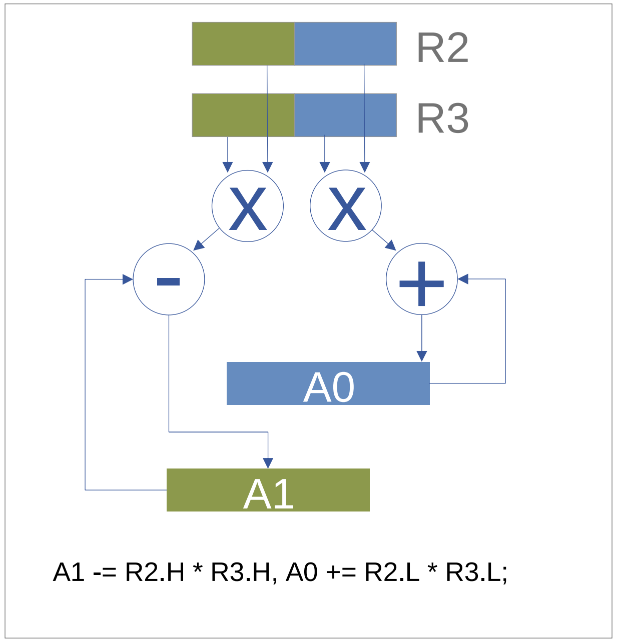


These operations effectively execute in a single cycle.

Dual MAC Operations—Example

Both MACs can be used at the same time to double the MAC throughput. The same two 32-bit input registers must be used (R2 and R3 in this example).

Dual MAC operation



A1 -= R2.H * R3.H, A0 += R2.L * R3.L;

These operations effectively execute in a single cycle.

NOTE: This can happen in parallel with a dual data fetch.

ANALOG DEVICES

Barrel Shifter

Enable shifting or rotating any number of bits within a 16-/32-/40-bit register in a single cycle

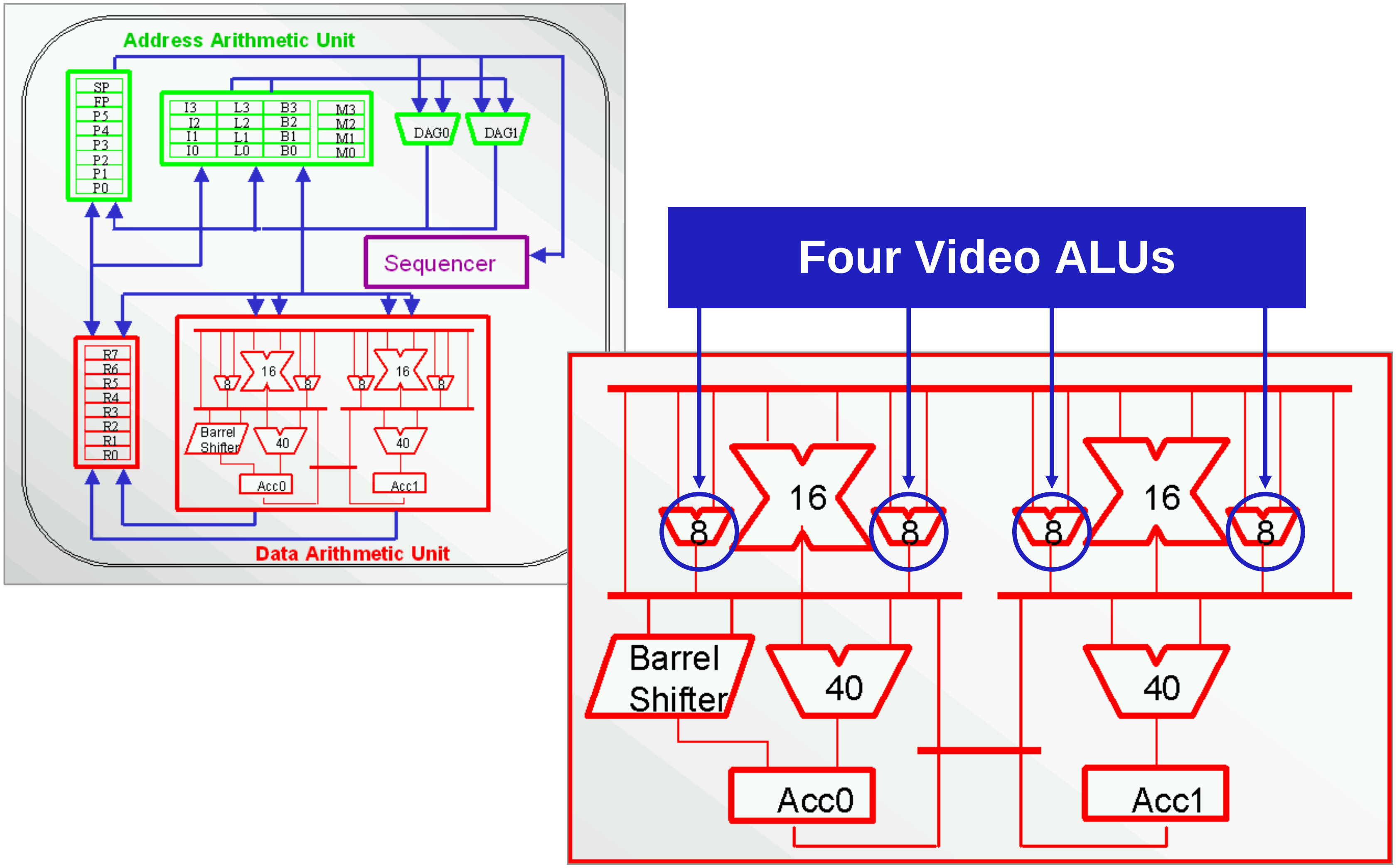
Perform individual bit operations on 32-bit data register contents

- **BITSET, BITCLR, BITTGL, BITTST**

Field Extract and Deposit instructions

- **Extract or insert a field of bits out of or into a 32-bit data register**

8-Bit Video ALUs

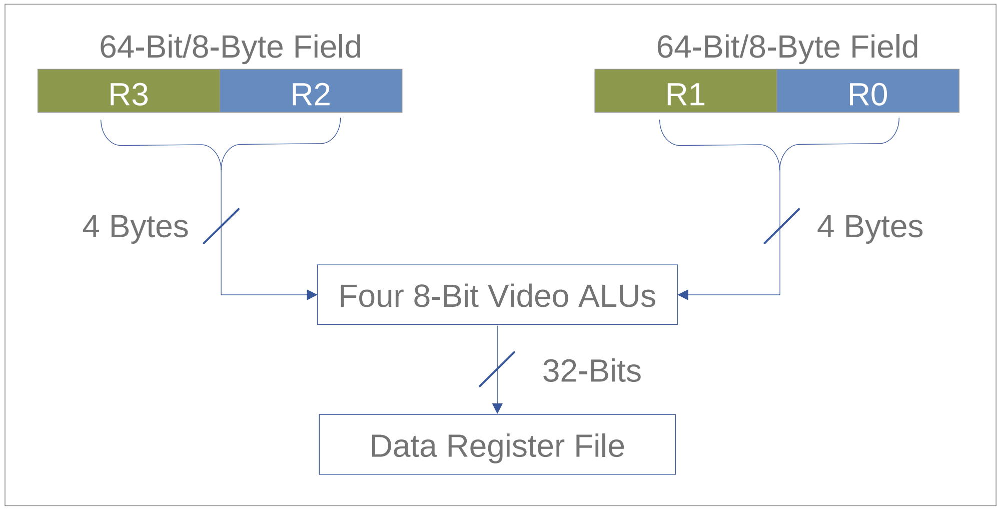


8-Bit ALU Operations

Four 8-bit ALUs provide parallel computational power targeted mainly for video operations.

- **Quad 8-bit add/subtract**
- **Quad 8-bit average**
- **SAA (Subtract-Absolute-Accumulate) instruction**

A quad 8-bit ALU instruction takes one cycle to complete.



Additional Arithmetic Instructions

There are a number of specialized instructions that are used to speed up the inner loop on various algorithms.

Bitwise XOR

- **Enable creating LFSR (Linear Feedback Shift Registers) for use in CRC calculations or the generation of PRN sequences**

Bit stream multiplexing, add on sign, compare select

- **Convolutional encoder and Viterbi decoder support**

Add/Subtract with prescale up/down

- **IEEE 1180–compliant 2D 8 x 8 DCTs (Discrete Cosine Transforms)**

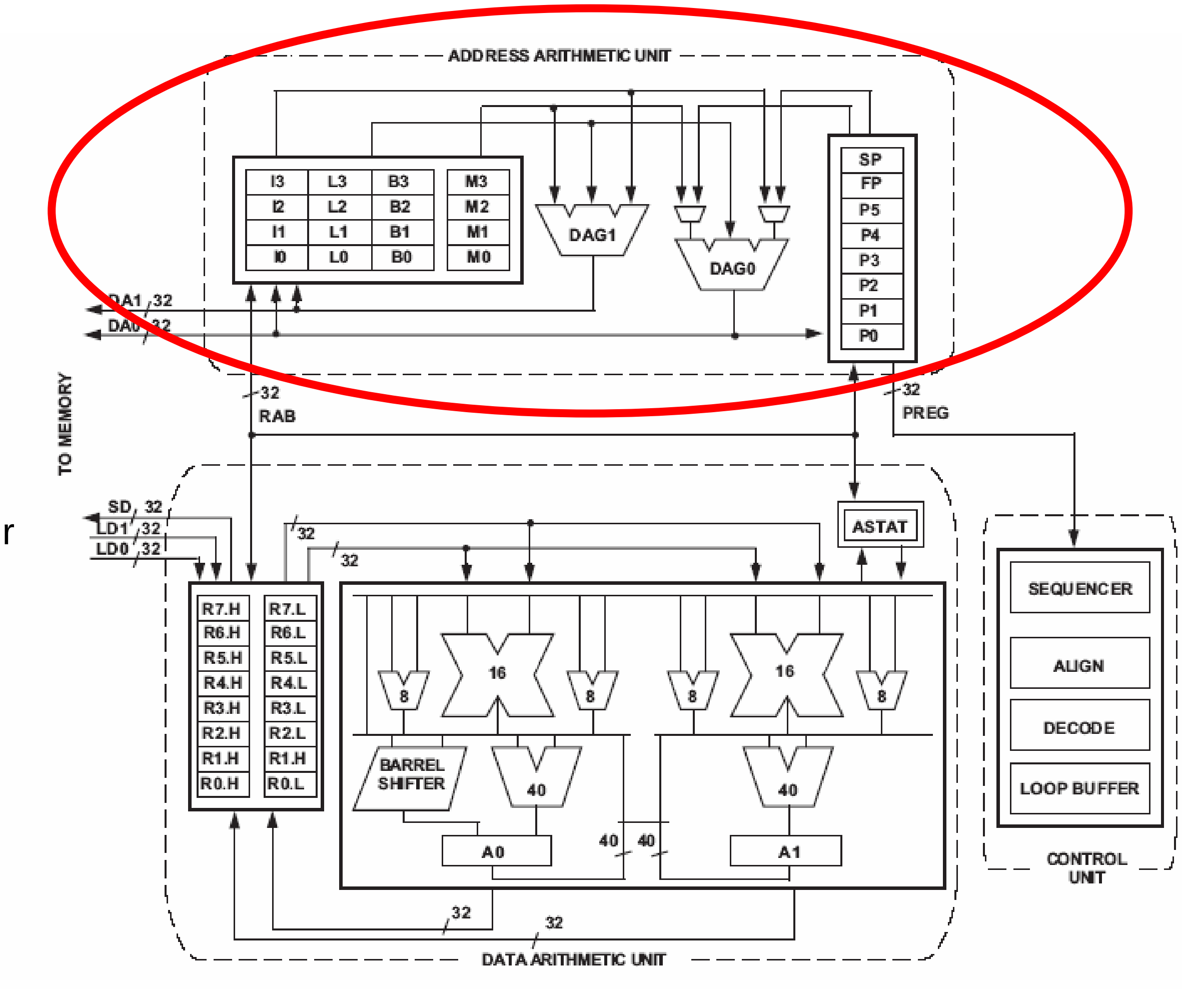
Vector search

- **Enable search a vector a pair at a time for greatest or least value**

The Addressing Unit

The addressing unit generates addresses for data fetches.

- **Two DAG (Data Address Generator) arithmetic units enable generation of independent 32-bit wide addresses that can reach anywhere within the Blackfin memory space.**
  - Up to two fetches can occur at the same time.



Address Registers

There are 6x general-purpose Pointer Registers.

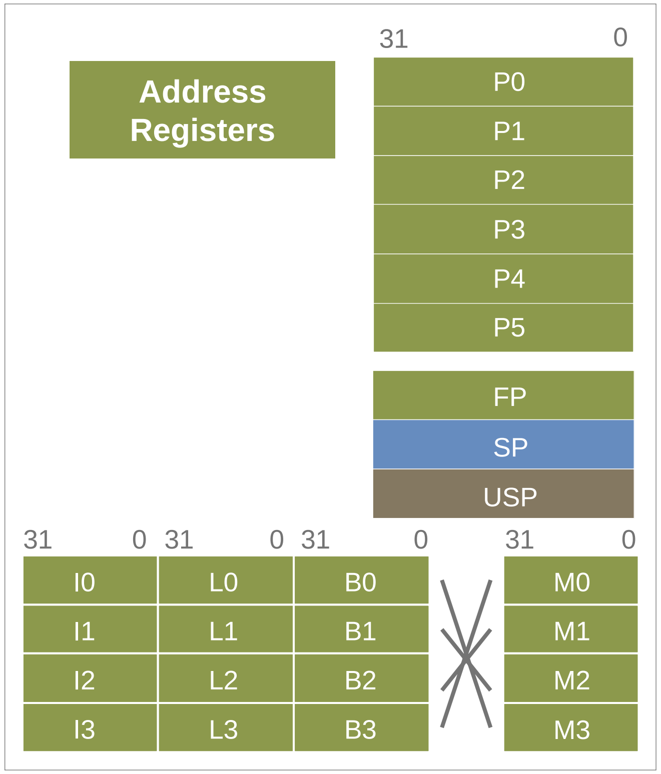
- **Used for GP 8-/16-/32-bit fetches from memory**

There are four sets of registers used for DSP-style data accesses.

- **Used for 16-/32-bit DSP data fetches such as dual data fetch, circular buffer addressing, and bit reversal**

There are also dedicated stack (SP) and frame (FP) pointers.

- **These are used for 32-bit accesses to stack frames.**



Pointer Registers: ***P0-P5*** **are referred to as "preg."**

Index Registers: ***I0-I3*** **are referred to as "ireg."**

Addressing

Addressing Unit supports:

- **Addressing only**
  - With specified Pointer or Index Register
- **Provide address and post modify**
  - Add an offset after the fetch is done
  - Circular buffering supported with this method
- **Provide address with an offset**
  - Add an offset before the fetch, but no pointer update
- **Update address only**
- **Modify address with reverse carry add**

All addressing is Register Indirect.

- **Index Registers I0-I3 (32-/16-bit accesses)**
- **Pointer Registers P0–P5 (32-/16-/8-bit accesses)**
- **Stack and Frame Pointer Registers (32-bit accesses)**

All addresses are Byte addresses.

- **Ordering is Little Endian.**
- **Addresses must be aligned for the word size being fetched.**
  - i.e., 32-bit fetches from addresses that are a multiple of four

Circular Buffer Example

Example

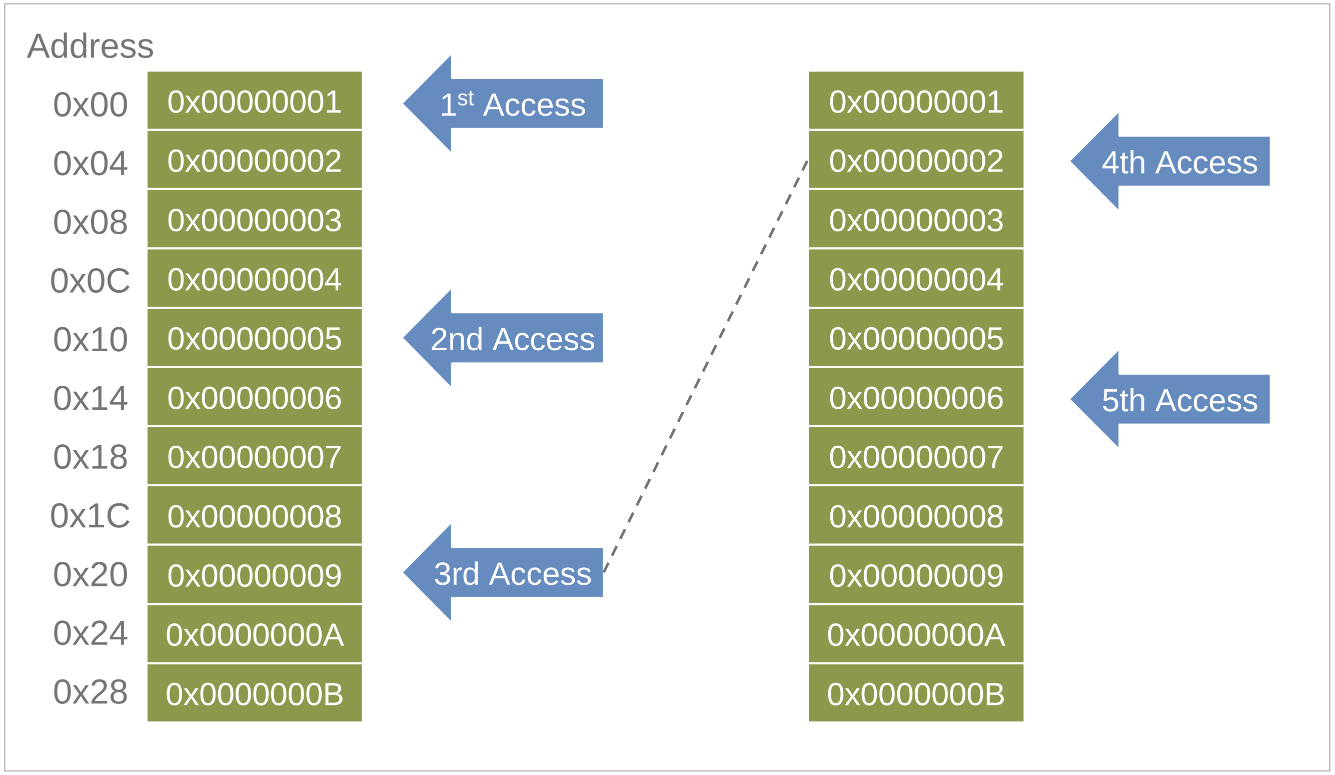
Base address (B) and Starting address (I) = 0

Buffer length
L = 44
(There are 11 data elements and each data element is 4-bytes)

Modify value M = 16
(4 elements * 4-bytes/element)

Example memory access:

R1 = [I0 ++ M2];



- **The Addressing Unit supports Circular Buffer pointer addressing.**
- **The process of boundary checking and pointer wrapping to stay in bounds happens in hardware with no overhead.**
- **Buffers can be placed anywhere in memory without restriction due to the Base address registers.**

The Sequencer

The sequencer's function is to generate addresses for fetching instructions.

- **Uses a variety of registers to select the next address**

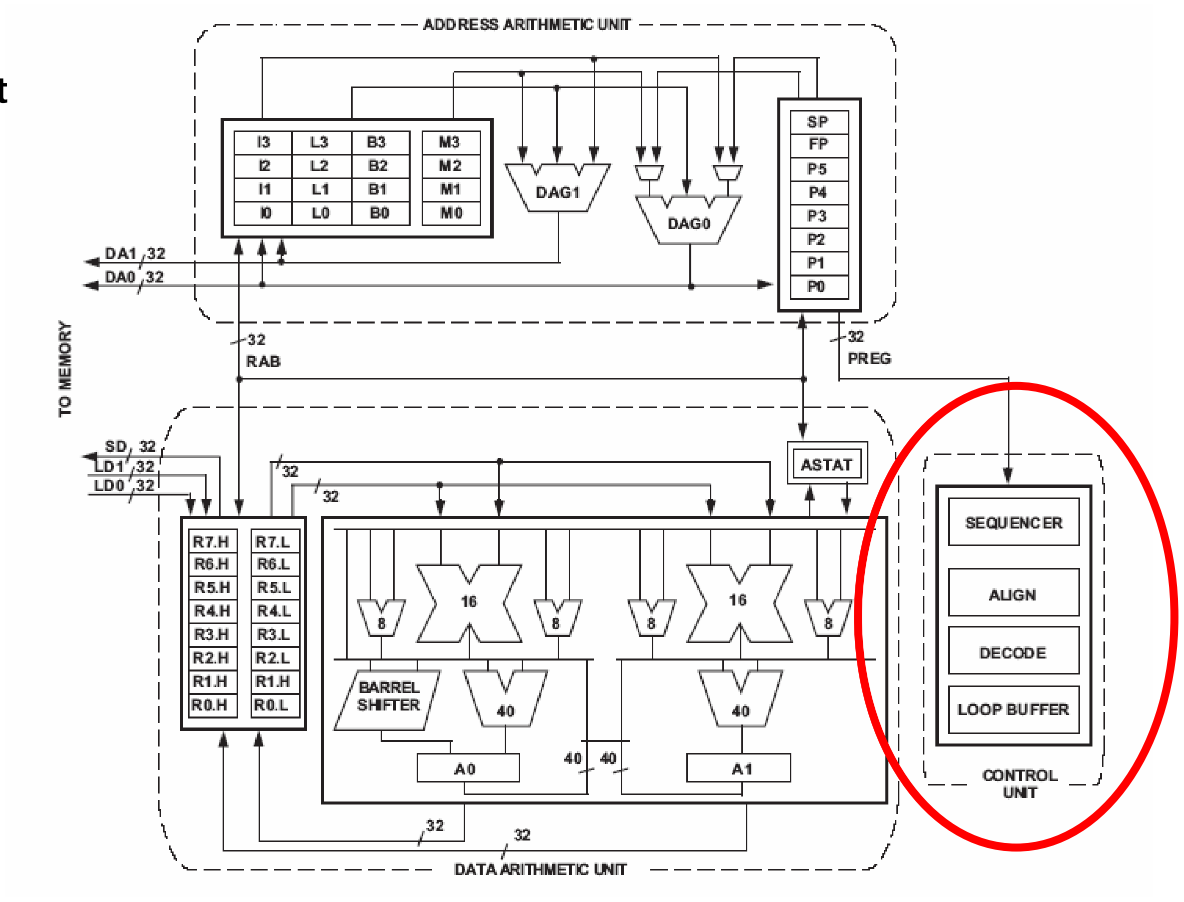
Aligns instructions as they are fetched

- **Always reads 64 bits from memory**
- **Realigns what is fetched into individual 16-/32-/64-bit opcodes before sending to the execution pipeline**

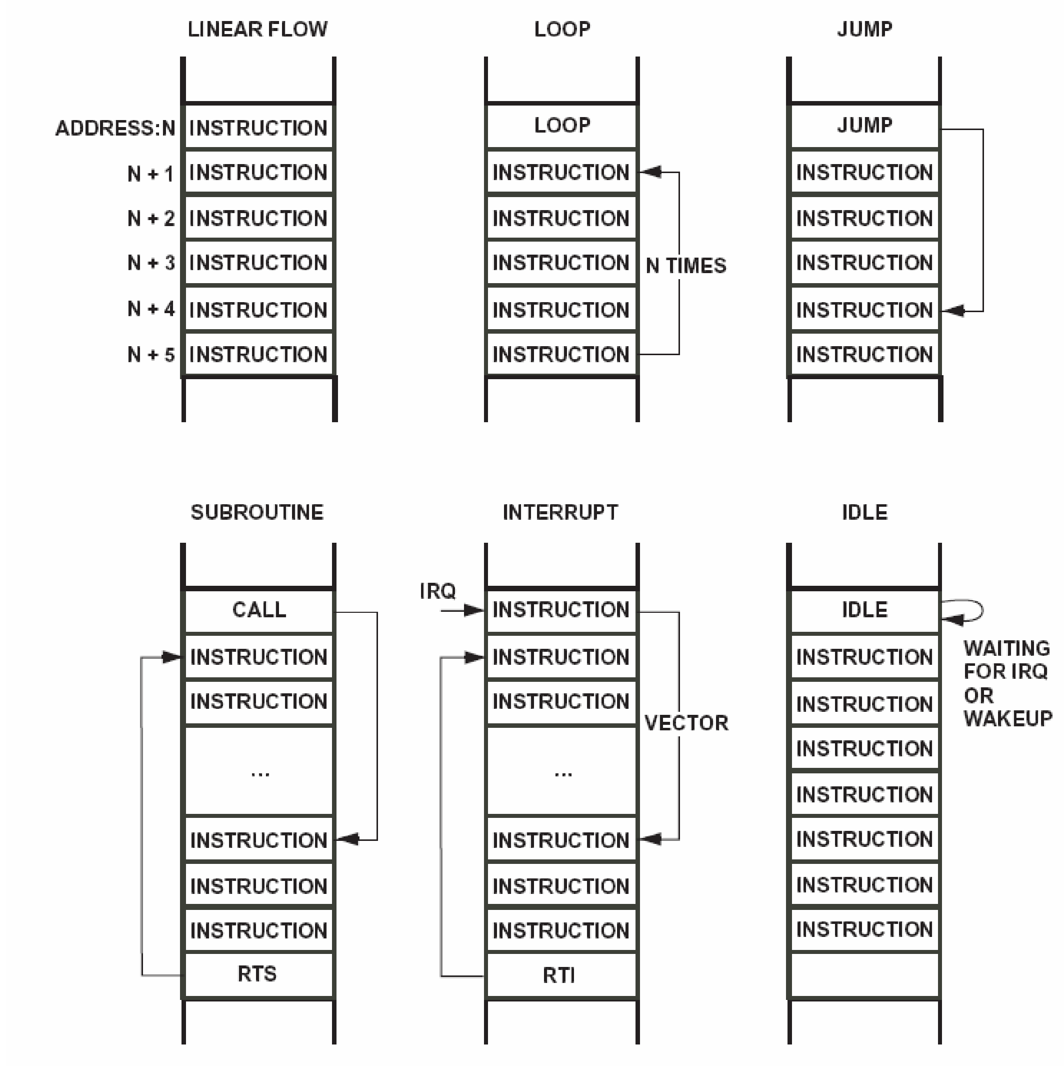
Handles events

- **Interrupts and exceptions**

Conditional execution



Program Flow



Sequencer Registers

These are registers that are used in the control of program flow.

Arithmetic Status (ASTAT) tracks status bits for core operations.

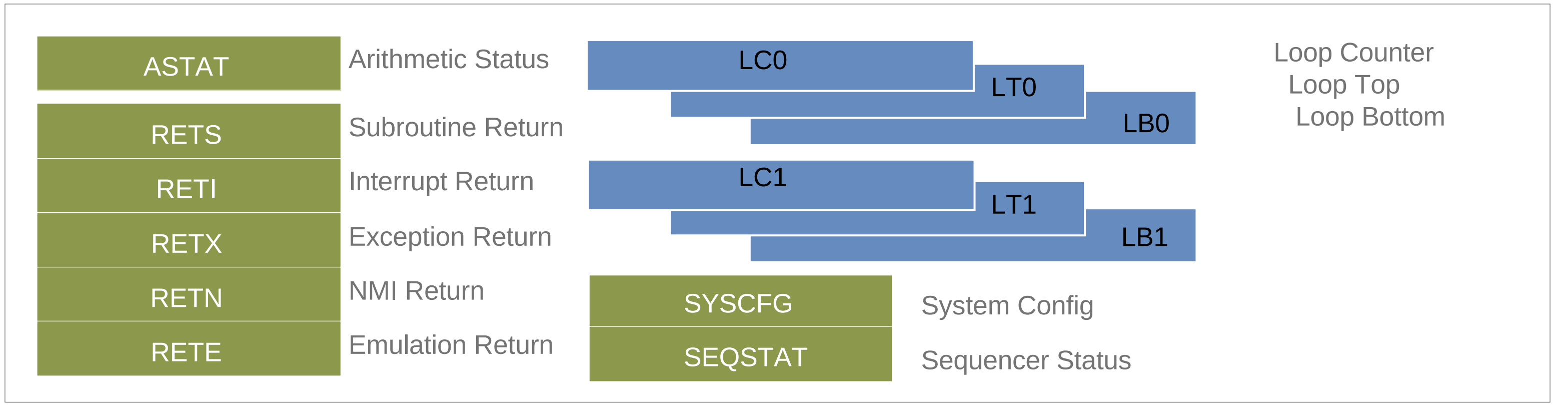
- **Used in conditional execution of instructions.**

Return address registers.

- **Hold the 32-bit return address for program flow interruptions.**

Two sets of hardware loop management registers.

- **They manage up to two nested levels of zero overhead looping.**
  - There are no core cycles spent managing the loop.
  - Looped code runs as efficiently as straight line code.



ANALOG DEVICES

Instruction Pipeline

The pipeline is fully interlocked. In the event of a data hazard, the sequencer automatically inserts stall cycles.

Pipeline Stage	Description
Instruction Fetch 1 (IF1)	Issue instruction address to IAB bus, start compare tag of instruction cache
Instruction Fetch 2 (IF2)	Wait for instruction data
Instruction Fetch 3 (IF3)	Read from IDB bus and align instruction
Instruction Decode (DEC)	Decode instructions
Address Calculation (AC)	Calculation of data addresses and branch target address
Data Fetch 1 (DF1)	Issue data address to DA0 and DA1 bus, start compare tag of data cache
Data Fetch 2 (DF2)	Read register files
Execute 1 (EX1)	Read data from LD0 and LD1 bus, start multiply and video instructions
Execute 2 (EX2)	Execute/Complete instructions (shift, add, logic, etc.)
Write Back (WB)	Writes back to register files, SD bus, and pointer updates (also referred to as the “commit” stage)

Instruction Pipeline

	IF1	IF2	IF3	DEC	AC	DF1	DF2	EX1	EX2	WB
Cycle										
N	INST #1									
N+1	INST #2	INST #1								
N+2	INST #3	INST #2	INST #1							
N+3	INST #4	INST #3	INST #2	INST #1						
N+4	INST #5	INST #4	INST #3	INST #2	INST #1					
N+5	INST #6	INST #5	INST #4	INST #3	INST #2	INST #1				
N+6	INST #7	INST #6	INST #5	INST #4	INST #3	INST #2	INST #1			
N+7	INST #8	INST #7	INST #6	INST #5	INST #4	INST #3	INST #2	INST #1		
N+8	INST #9	INST #8	INST #7	INST #6	INST #5	INST #4	INST #3	INST #2	INST #1	
N+9	INST #10	INST #9	INST #8	INST #7	INST #6	INST #5	INST #4	INST #3	INST #2	INST #1
N+10	INST #11	INST #10	INST #9	INST #8	INST #7	INST #6	INST #5	INST #4	INST #3	INST #2
N+11	INST #12	INST #11	INST #10	INST #9	INST #8	INST #7	INST #6	INST #5	INST #4	INST #3
N+12	INST #13	INST #12	INST #11	INST #10	INST #9	INST #8	INST #7	INST #6	INST #5	INST #4

Once the pipe is filled (e.g., at cycle N+9 in this case), one instruction will exit the pipeline on each core clock tick.

- **i.e., effectively, one instruction per core clock cycle executed**

Blackfin Event Handling

Blackfin events include:

- **Interrupts**
  - Generated by hardware (e.g., DMA complete) or software
- **Exceptions**
  - Error condition or service related

Handling is split between the CEC (Core Event Controller) and SIC (System Interrupt Controller).

- **CEC has 16 levels and deals with requests on a priority basis.**
  - The nine lowest levels are general purpose and are used for Peripheral Interrupt Requests.
  - Each level has a 32-bit Interrupt Vector Register that points to the start of the ISR for that level.
- **SIC allows enabling Peripheral Interrupt Requests and mapping to specific CEC GP levels.**
  - Allows setting peripheral request priorities

System and Core Interrupts (ADSP-BF533 example)

System Interrupt Source	IVG #[1]
PLL Wakeup interrupt	IVG7
DMA error (generic)	IVG7
PPI error interrupt	IVG7
SPORT0 error interrupt	IVG7
SPORT1 error interrupt	IVG7
SPI error interrupt	IVG7
UART error interrupt	IVG7
RTC interrupt	IVG8
DMA 0 interrupt (PPI)	IVG8
DMA 1 interrupt (SPORT0 RX)	IVG9
DMA 2 interrupt (SPORT0 TX)	IVG9
DMA 3 interrupt (SPORT1 RX)	IVG9
DMA 4 interrupt (SPORT1 TX)	IVG9
DMA 5 interrupt (SPI)	IVG10
DMA 6 interrupt (UART RX)	IVG10
DMA 7 interrupt (UART TX)	IVG10
Timer0 interrupt	IVG11
Timer1 interrupt	IVG11
Timer2 interrupt	IVG11
PF interrupt A	IVG12
PF interrupt B	IVG12
DMA 8/9 interrupt (MemDMA0)	IVG13
DMA 10/11 interrupt (MemDMA1)	IVG13
Watchdog timer interrupt	IVG13

Event Source	IVG #	Core Event Name
Emulator	0	EMU
Reset	1	RST
Non Maskable Interrupt	2	NMI
Exceptions	3	EVSW
Reserved	4	-
Hardware Error	5	IVHW
Core Timer	6	IVTMR
General Purpose 7	7	IVG7
General Purpose 8	8	IVG8
General Purpose 9	9	IVG9
General Purpose 10	10	IVG10
General Purpose 11	11	IVG11
General Purpose 12	12	IVG12
General Purpose 13	13	IVG13
General Purpose 14	14	IVG14
General Purpose 15	15	IVG15

Highest

PRIORITY

Lowest

[1] Note: Default IVG configuration shown.

Variable Instruction Lengths

The Blackfin architecture uses three instruction opcode lengths to obtain the best code density while maintaining high performance.

16-bit instructions

- **Most control-type instructions and data fetches are 16-bits long to improve code density.**

32-bit instructions

- **Most control-type instructions with an immediate value in the expression and most arithmetic instructions are 32-bits in length.**

Multi-issue 64-bit instructions

- **Certain 32-bit instructions can be executed in parallel with a pair of specific 16-bit instructions and specified with one 64-bit instruction.**
  - Typically a 32-bit ALU/MAC instruction and one or two data fetch instructions
- **The delimiter symbol to separate instructions in a multi-issue instruction is a double pipe character "||."**

Example:

```
A1+=R0.L*R1.H, A0+=R0.L*R1.L || r0 = [i0++] || r1 = [i1++];
```

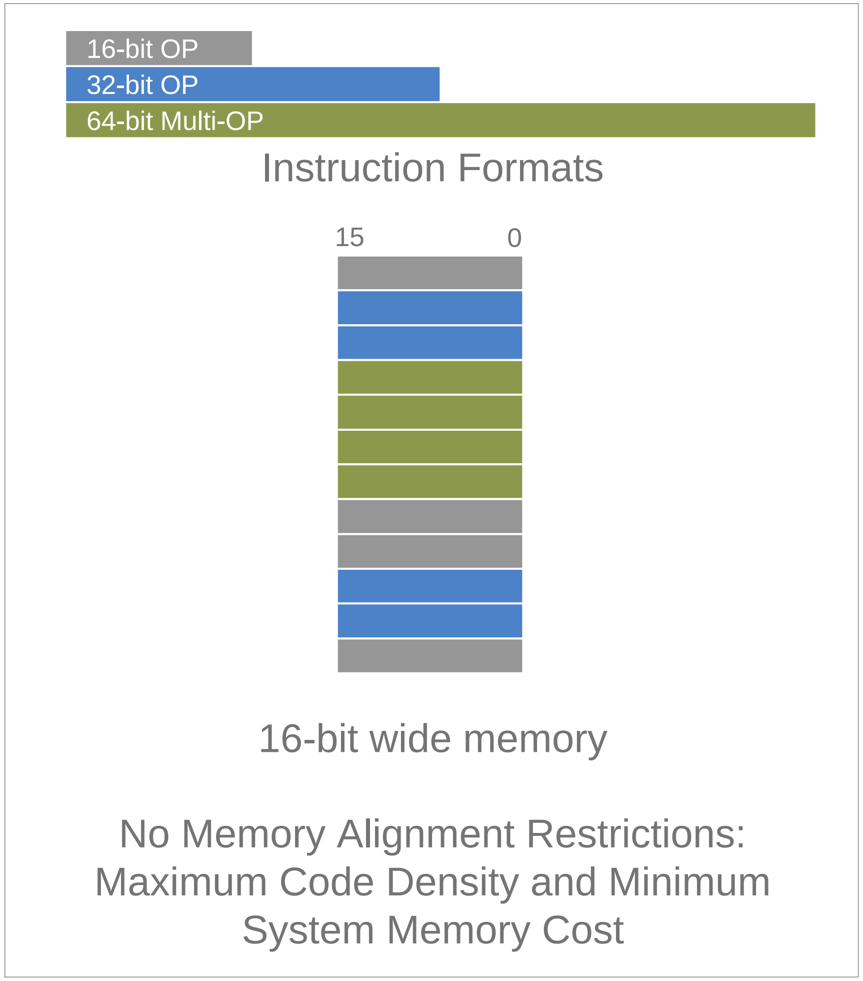
Instruction Packing

When code is compiled and linked, the instructions are packed into memory as densely as possible.

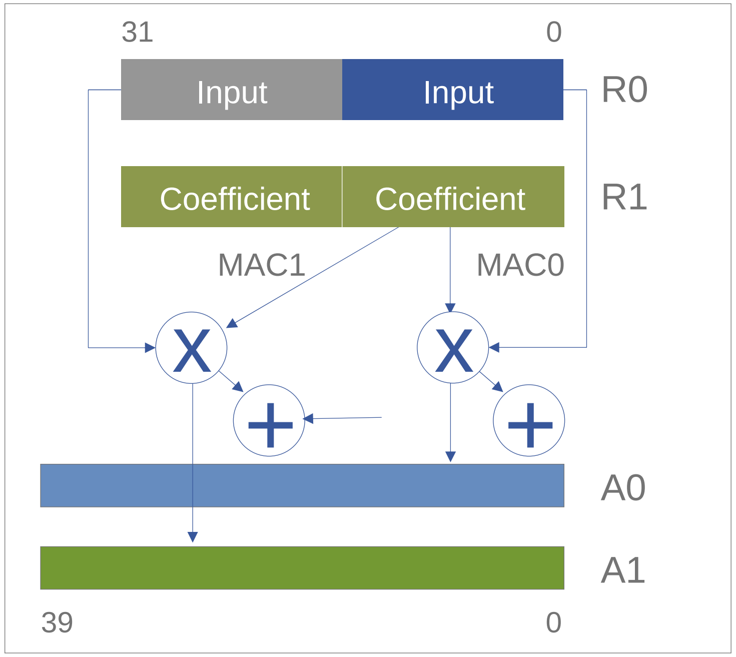
- **i.e., no wasted memory space**

No memory alignment restrictions for code:

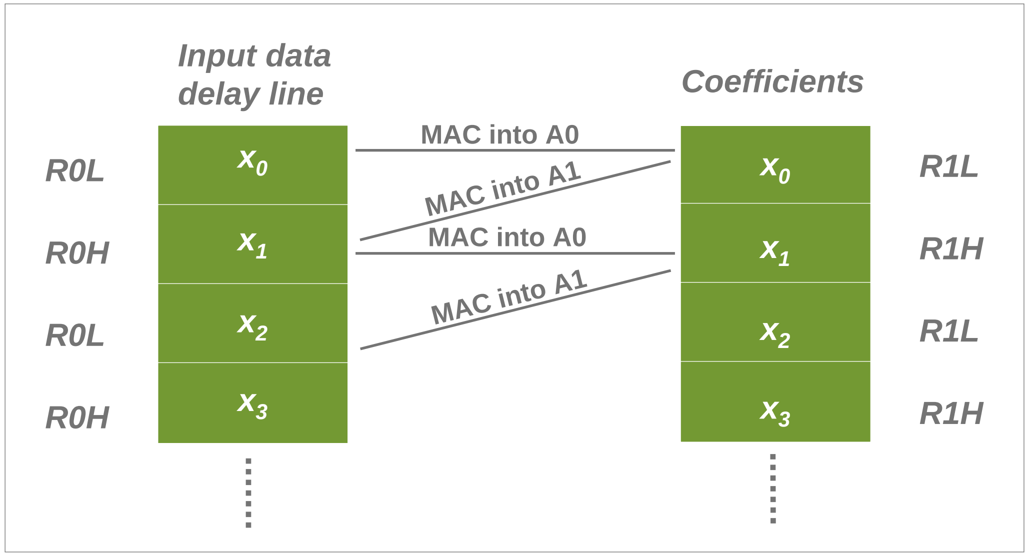
- **Instructions can be placed anywhere in memory.**
- **The sequencer fetches 64-bits of instruction at a time (from 8-byte boundaries) and performs an alignment operation to:**
  - Isolate shorter opcodes
  - Realign larger opcodes that straddle 4-/8-byte boundaries
- **This realignment hardware is transparent to the user.**


16-bit OP
32-bit OP
64-bit Multi-OP
Instruction Formats
15
0
16-bit wide memory
No Memory Alignment Restrictions: Maximum Code Density and Minimum System Memory Cost

16-bit FIR Filter Example—0.5 cycles per tap



$$y(k)=\sum_{j=0}^{M-1}c(j)x(k-j),\quad 0\le k<N$$



```
Loop:    A1+=R0.H*R1.L, A0+=R0.L*R1.L  || R0.L = [I0++] ||  nop;
Loopend: A1+=R0.L*R1.H, A0+=R0.H*R1.H || R0.H = [I0++] ||  R1 = [I1++];
```

Samples in R0

Coefficients in R1

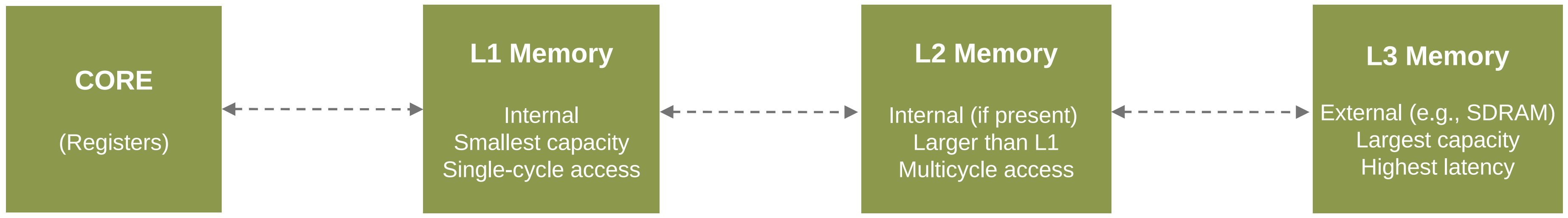
- Performs operations in support of two filter outputs in each clock cycle

Bus and Memory Architecture

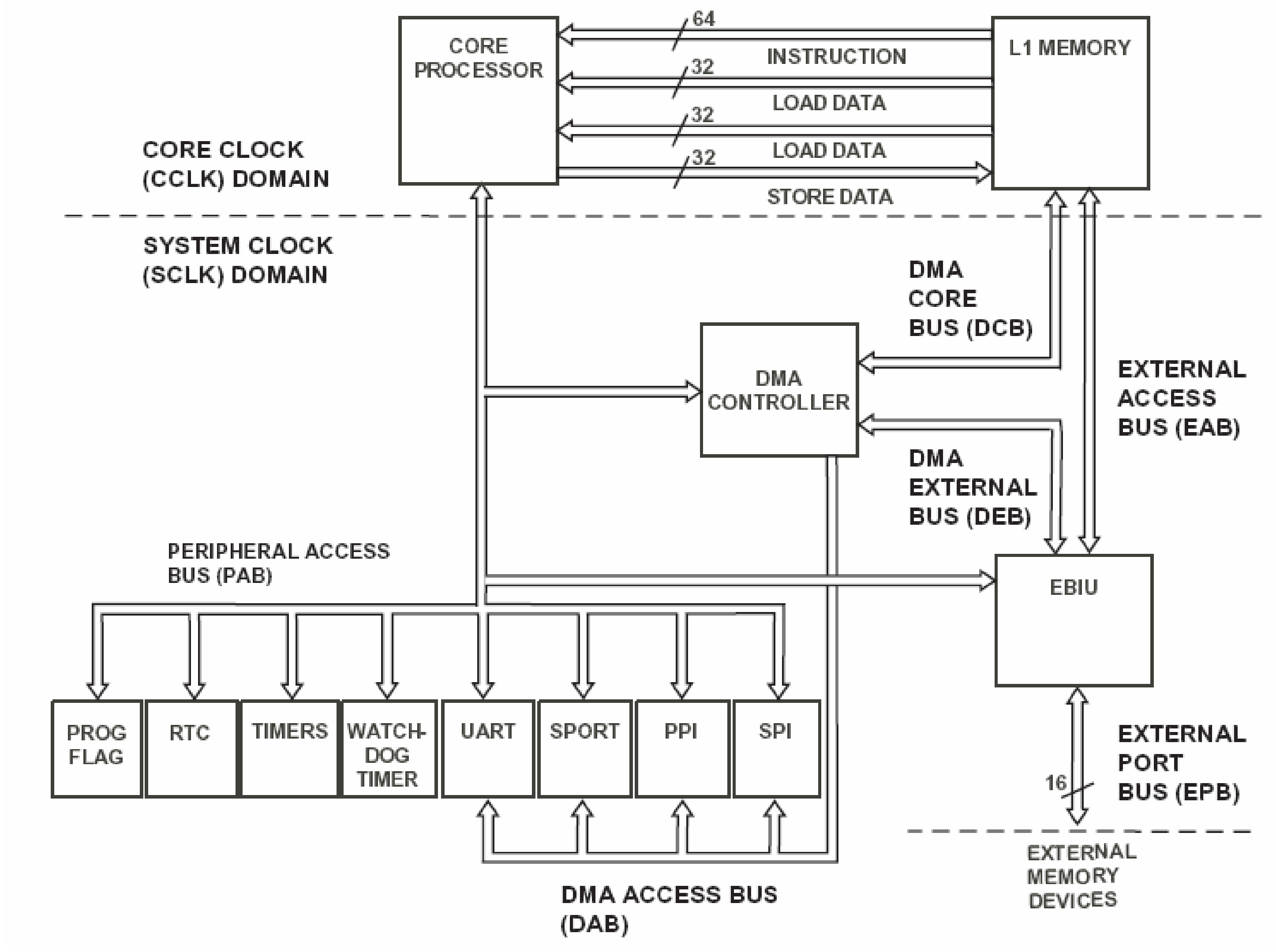
Blackfin Memory Hierarchy

The Blackfin architecture 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., L3).

- **Portions of L1 can be configured as cache, which allows increased memory performance by prefetching and storing copies of code/data from L2/L3.**



Internal Bus Structure of the ADSP-BF533



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–the Blackfin processor supports both.

- **Microcontrollers 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 (e.g. code overlays), as they need direct control over which code runs in the fastest memory.**

The Blackfin processor allows the programmer to choose one or both methods to optimize system performance.

- **Portions of L1 instruction and data memory can be configured to be used as SRAM or as cache.**
  - Enabling these 16K Byte blocks for use as cache reduces the amount of L1 available as SRAM. However, there is still space available for code, such as critical functions or interrupt handlers.

Cache and Memory Management

Cache allows users to take advantage of single-cycle memory without having to specifically move instructions and or data "manually."

- **L2/L3 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.**

Cache automatically optimizes code and data by keeping recently used information in L1.

- **LRU (Least Recently Used) algorithm is used for determining which cache line to replace.**

Cache is managed by the memory management unit though a number of CPLBs (Cachability, Protection, and Lookaside Buffers).

- **The CPLBs divide up the Blackfin memory space into pages and allow individual page control of memory management items such as:**
  - User/Supervisor Access Protection
  - Read/Write Access Protection
  - Cacheable or Non-Cacheable

Additional Blackfin Core Features

Direct Memory Access (DMA)

The Blackfin processors includes a DMA (Direct Memory Access) facility on all CPUs.

- **Enable moving data between memory and a peripheral, or memory to memory without using any core cycles**
  - Core only involved in setting up DMA parameters
- **Core is interrupted when DMA completes, thereby improving efficiency of data flow operations.**
  - Alternatively, DMA status allows polling operations.
- **The DMA controller has dedicated buses to connect between the DMA unit itself and:**
  - Peripherals
  - External memory (e.g., L3 or SDRAM)
  - Core memory (e.g., L1)

Power Management Options

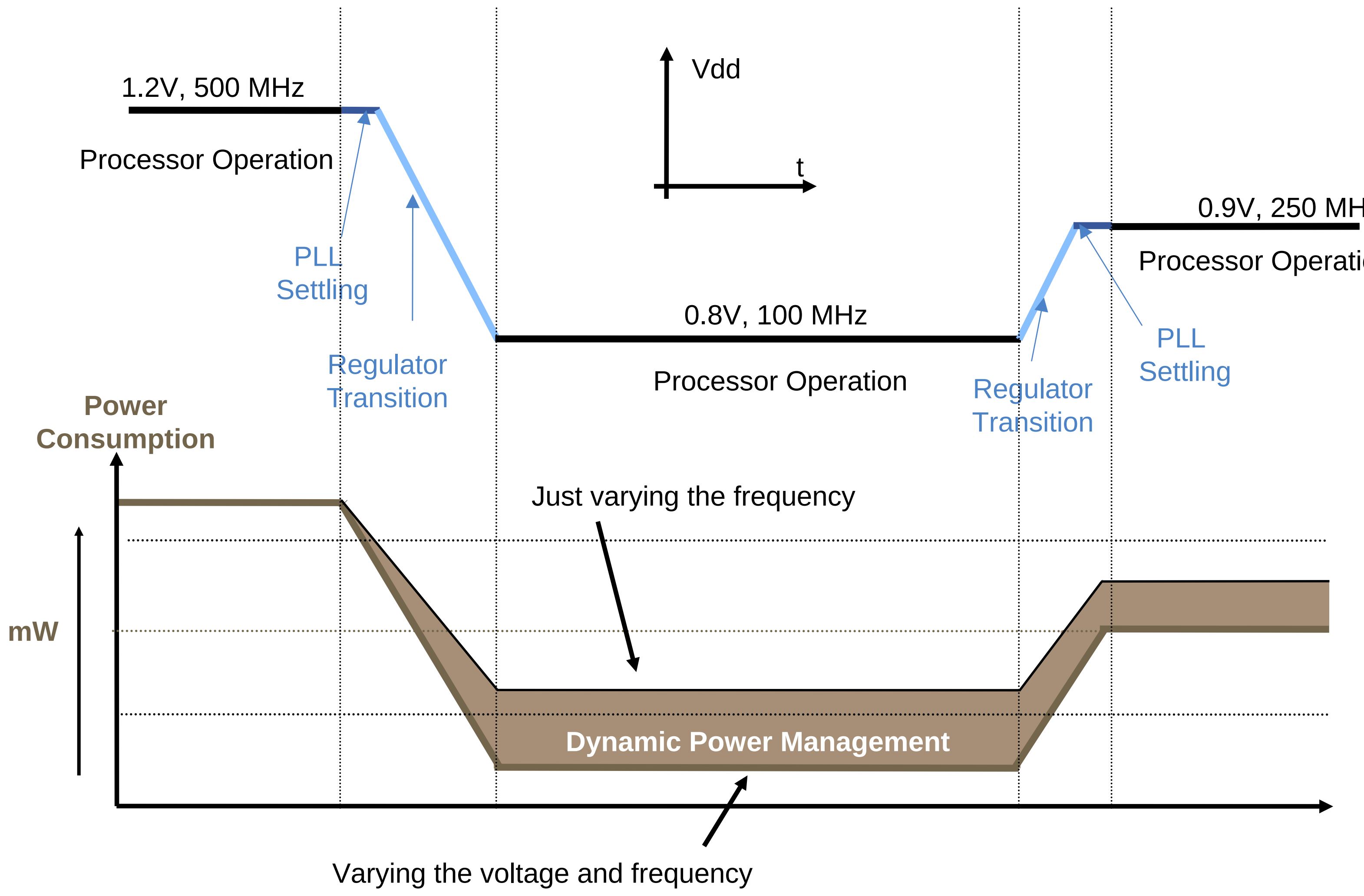
Low active power

- **Flexible power management with automatic power-down for unused peripheral sections.**
- **Dynamic power management allows dynamic modification of both frequency and voltage.**
  - PLL can multiply CLKIN from 1x to 64x.
  - Core voltage can be optimized for the operating frequency.

Low standby power

- **5 power modes**
  - Full on, active, sleep, deep sleep, hibernate
- **Real-time clock with alarm and wakeup features**
  - RTC has its own power supply and clock.
  - It is unaffected by hardware or software reset.

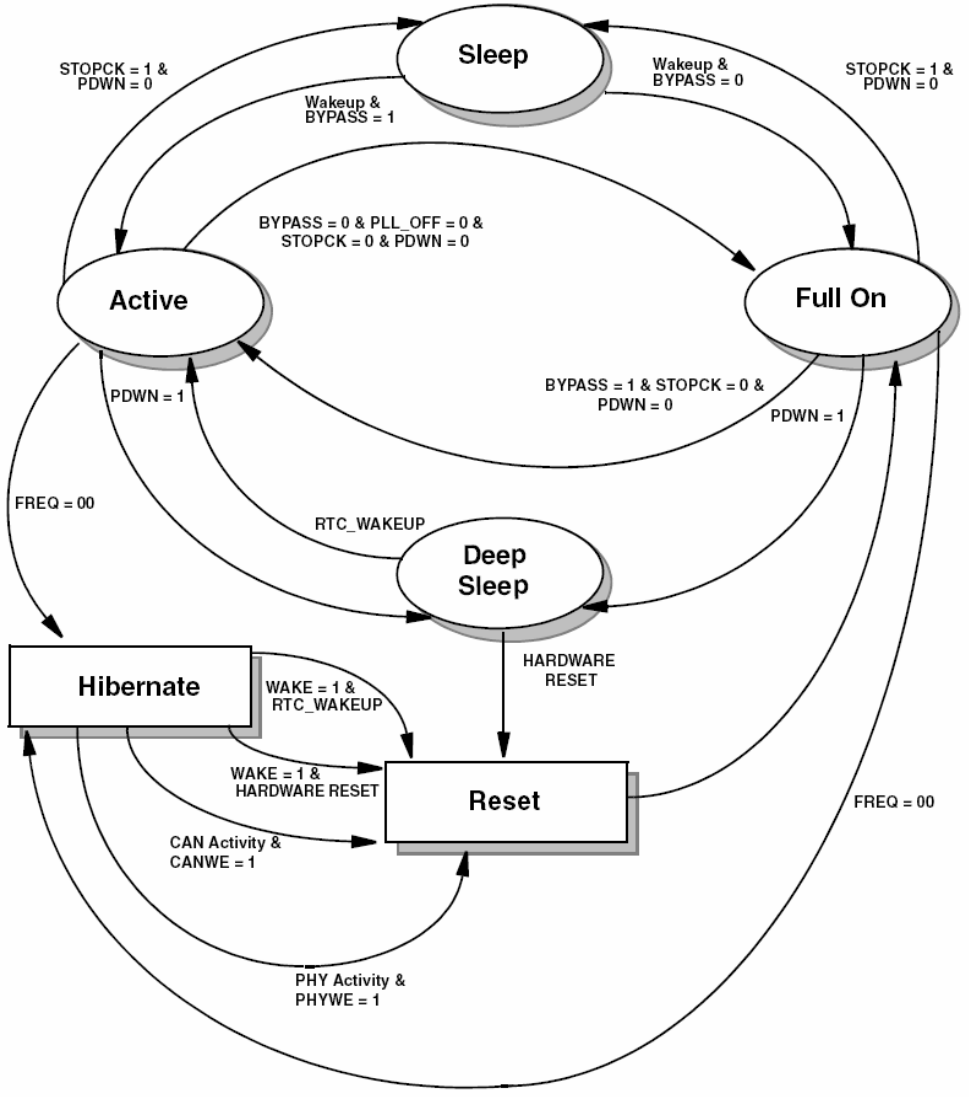
Blackfin Processors Optimize Power Consumption



Power Mode Transitions

Under software control, the application can change from one power state to another.

A set of libraries (i.e., System Services) enable control of clocking, core voltage, and power states with just a function call.



Hardware Debug Support

Advanced Support for Embedded Debug

The JTAG port is also used to provide in-circuit emulation capabilities.

- Nonintrusive debugging
- BTC (background telemetry channel)

Hardware breakpoints

- Six instruction and two data watchpoints

Performance monitor unit

- Counters for cycles and occurrences of specific activities

Execution trace buffer

- Stores last 16 nonincremental PC values

ANALOG DEVICES

Summary

The Blackfin core has a number of features and instructions that support both control and DSP types of operations.

The Blackfin's high performance memory/bus architecture supports zero wait state instruction and dual fetches from L1 at the core clock rate, all at the same time.

- Large applications that reside in the larger but slower external memory still benefit from the high performance L1 memory through the use of caching and/or dynamic downloading

The Blackfin architecture enables both efficient implementation of media-related applications and efficient development of those applications.

Resources

For detailed information on the Blackfin architecture, please refer to the Analog Devices website which has links to 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.

Kaztek Systems provides worldwide technical training on processor architecture and systems development using Analog Devices Processors. For more information

Visit www.kaztek.com or Email info@kaztek.com