UIUC CS 433

Analog Devices Incorporated Blackfin

CS 433

Prof. Luddy Harrison

Processor Presentation Series

Note on this presentation series

- These slide presentations were prepared by students of CS433 at the University of Illinois at Urbana-Champaign
- All the drawings and figures in these slides were drawn by the students. Some drawings are based on figures in the manufacturer's documentation for the processor, but none are electronic copies of such drawings
- You are free to use these slides provided that you leave the credits and copyright notices intact

Presentation Outline

- Introduction
- Architecture Overview
- Processor Core
- Signal Chain
- Instruction Set
- Blackfin Platforms
- Targeted Application
- References

Introduction

- New type of 16-32-bit embedded processor
- Combine RISC MCU and DSP functionalities
- Specifically designed for the computational demands and power constraints for embedded audio, video and communications applications
- Micro Signal Architecture (MSA) based 32-bit RISC-like instruction set and dual 16-bit multiply accumulate (MAC) signal processing and 8-bit video processing
- Performs equally well in both signal processing and control processing applications

Introduction

- Advanced memory management that supports memory-protected and non memory-protected embedded operating systems
- Supported by ADI's CROSSCORE development tool chain, which includes the VisualDSP++ Integrated Development and Debugging Environment (IDDE), and by Green Hills' MULTI IDE tool suite

Introduction

- **The Blackfin Family of Processors**
  - ADSP-BF535 was the first released
  - followed in March 2003 by three pin-compatible devices, the ADSP-BF531, ADSP-BF532, and ADSP-BF533
  - In January of 2005, Analog Devices introduced three Blackfin processors with embedded connectivity: ADSP-BF536, ADSP-BF537, and ADSP-BF534
  - dual-core symmetric multiprocessor, the ADSP-BF561

Blackfin Processor Specifications

Feature	ADSP-BF535	ADSP-BF531	ADSP-BF532	ADSP-BF533	ADSP-BF561	ADSP-BF536	ADSP-BF537	ADSP-BF534
Max Clock Speed (MHz)	350	400	400	750	600	400	600	500
Memory (Kbytes)	308	52	84	148	328	100	132	132
External Memory (bus)	32-bit	16-bit	16-bit	16-bit	32-bit	16-bit	16-bit	16-bit
Parallel Peripheral Interface	No	Yes	Yes	Yes	Yes (2)	Yes	Yes	Yes

Blackfin Processor Specifications

Feature	ADSP-BF535	ADSP-BF531	ADSP-BF532	ADSP-BF533	ADSP-BF561	ADSP-BF536	ADSP-BF537	ADSP-BF534
UARTs, Timers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SPORTs, SPI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Programmable Flags	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TWI-Compatible	No	No	No	No	No	Yes	Yes	Yes

UIUC CS 433

Blackfin Processor Specifications

Feature	ADSP-BF535	ADSP-BF531	ADSP-BF532	ADSP-BF533	ADSP-BF561	ADSP-BF536	ADSP-BF537	ADSP-BF534
Watchdog Timer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RTC	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Core Voltage (V)	1-1.6	0.8-1.2	0.8-1.2	0.8-1.4	0.8-1.2	0.8-1.2	0.8-1.2	0.8-1.2
Core Voltage Regulation	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Blackfin Processor Specifications

Feature	ADSP-BF535	ADSP-BF531	ADSP-BF532	ADSP-BF533	ADSP-BF561	ADSP-BF536	ADSP-BF537	ADSP-BF534
Package Size	260 PBGA	160 Mini-BGA, 176 LQFP, 169 Sparse PBGA	160 Mini-BGA, 176 LQFP, 169 Sparse PBGA	160 Mini-BGA, 169 Sparse PBGA	256 Mini-BGA, 297 PBGA	182 Mini-BGA, 208 Sparse Mini-BGA	182 Mini-BGA, 208 Sparse Mini-BGA	182 Mini-BGA, 208 Sparse Mini-BGA
Lead-Free Package Option	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Recommended Operating Conditions

Parameter		Min	Nor	Max	Unit
V_{DDINT}	Internal Supply Voltage (ADSP-BF531 and ADSP-BF532)	0.8	1.2	1.32	V
V_{DDINT}	Internal Supply Voltage (ADSP-BF533)	0.8.	1.26	1.32	V
V_{DDENT}	External Supply Voltage	2.25	2.5	3.6	V
V_{DDRTC}	Real-Time Clock Power Supply Voltage	2.25		3.6	V
V_{IH}	High Level Input Voltage1, 2 @VDDEXT =maximum	2.0		3.6	V
$V_{IHCLKIN}$	High Level Input Voltage3 @ VDDEXT =maximum	2.2		3.6	V
V_{IL}	Low Level Input Voltage2, 4 @ VDDEXT =minimum	-0.3		0.6	V

Electrical Characteristics

Parameters		Test Condition	Min	Max	Unit
VOH	High Level Output Voltage	@ VDDEXT = 3.0V, IOH = –0.5 mA	2.4		V
VOL	Low Level Output Voltage	@ VDDEXT = 3.0V, IOL = 2.0 mA		0.4	V
IIH	High Level Input Current	@ VDDEXT = maximum, VIN = VDD maximum		10	uA
IIHP	High Level Input Current JTAG	@ VDDEXT = maximum, VIN = VDD maximum		20	uA
IIL	Low Level Input Current	@ VDDEXT = maximum, VIN = 0 V		10	uA
IOZH	Three-State Leakage Current	@ VDDEXT = maximum, VIN = VDD maximum		10	uA
IOZL	Three-State Leakage Current	@ VDDEXT = maximum, VIN = 0 V		10	uA
IIN	Input Capacitance	fIN = 1 MHz, TAMBIENT = 25°C, VIN = 2.5 V		8	pF

Pin Descriptions

Pin Name	I/O	Function	Driver Type
ADDR19–1	O	Address Bus for Async/Sync Access	A2
DATA15–0	I/O	Data Bus for Async/Sync Access	A2
ABE1–0/SDQM1–0	O	Byte Enables/Data Masks for Async/Sync Access	A2
BR3	I	Bus Request	A2
BG	O	Bus Grant	A2
BGH	O	Bus Grant Hang	A2
AMS3–0	O	Bank Select	A2
ARDY	I	Hardware Ready Control	
AOE	O	Output Enable	A2

Pin Descriptions

Pin Name	I/O	Function	Driver Type
ARE	O	Read Enable	A2
AWE	O	Write Enable	A2
SRAS	O	Row Address Strobe	A2
SCAS	O	Column Address Strobe	A2
SWE	O	Write Enable	A2
SCKE	O	Clock Enable	A2
CLKOUT	O	Clock Output	B4
SA10	O	A10 Pin	A2
SMS	O	Bank Select	A2

Pin Descriptions

Pin Name	I/O	Function	Driver Type
TMR0	I/O	Timer 0	C5
TMR1/*PPI_FS1*	I/O	Timer 1/*PPI Frame Sync1*	C5
TMR2/*PPI_FS2*	I/O	Timer 2/*PPI Frame Sync2*	C5
PF0/*SPISS*	I/O	Programmable Flag 0/*SPI Slave Select Input*	C5
PF1*/SPISEL1/TM RCLK*	I/O	Programmable Flag 1/*SPI Slave Select Enable 1/External Timer Reference*	C5
PF2/*SPISEL2*	I/O	Programmable Flag 2/*SPI Slave Select Enable 2*	C5
PF3/*SPISEL3/PPI _FS3*	I/O	Programmable Flag 3/*SPI Slave Select Enable 3/PPI Frame Sync 3*	C5

Pin Descriptions

Pin Name	I/O	Function	Driver Type
PF4/*SPISEL4/PPI 15*	I/O	Programmable Flag 4/*SPI Slave Select Enable 4 / PPI 15*	C5
PF5/*SPISEL5/PPI 14*	I/O	Programmable Flag 5/*SPI Slave Select Enable 5 / PPI 14*	C5
PF6/*SPISEL6/PPI 13*	I/O	Programmable Flag 6/*SPI Slave Select Enable 6 / PPI 13*	C5
PF7/*SPISEL7/PPI 12*	I/O	Programmable Flag 7/*SPI Slave Select Enable 7 / PPI 12*	C5
PF8/*PPI11*	I/O	*Programmable Flag 8/PPI 11*	C5
PF9/*PPI10*	I/O	Programmable Flag 9/*PPI 10*	C5
PF10/*PPI9*	I/O	Programmable Flag 10/*PPI 9*	C5

Pin Descriptions

Pin Name	I/O	Function	Driver Type
PF11/*PPI8*	I/O	Programmable Flag 11/*PPI 8*	C5
PF12/*PPI7*	I/O	Programmable Flag 12/*PPI 7*	C5
PF13/*PPI6*	I/O	Programmable Flag 13/*PPI 6*	C5
PF14/*PPI5*	I/O	Programmable Flag 14/*PPI 5*	C5
PF15/*PPI4*	I/O	Programmable Flag 15/*PPI 4*	C5
PPI3–0	I/O	PPI3–0	C5
PPI_CLK	I	PPI Clock	C5
RSCLK0	I/O	SPORT0 Receive Serial Clock	D6
RFS0	I/O	SPORT0 Receive Frame Sync	C5
DR0PRI	I	SPORT0 Receive Data Primary	

Pin Descriptions

Pin Name	I/O	Function	Driver Type
DR0SEC	I	SPORT0 Receive Data Secondary	
TSCLK0	I/O	SPORT0 Transmit Serial Clock	D6
TFS0	I/O	SPORT0 Transmit Frame Sync	C5
DT0PRI	O	SPORT0 Transmit Data Primary	C5
DT0SEC	O	SPORT0 Transmit Data Secondary	C5
RSCLK1	I/O	SPORT1 Receive Serial Clock	D6
RFS1	I/O	SPORT1 Receive Frame Sync	C5
DR1PRI	I	SPORT1 Receive Data Primary	
DR1SEC	I	SPORT1 Receive Data Secondary	
TSCLK1	I/O	SPORT1 Transmit Serial Clock	D6

Pin Descriptions

Pin Name	I/O	Function	Driver Type
TFS1	I/O	SPORT1 Transmit Frame Sync	C5
DT1PRI	O	SPORT1 Transmit Data Primary	C5
DT1SEC	O	SPORT1 Transmit Data Secondary	C5
MOSI	I/O	Master Out Slave In	C5
MISO	I/O	Master In Slave Out	C5
SCK	I/O	SPI Clock	D6
RX	I	UART Receive	
TX	O	UART Transmit	C5
RTXI	I	RTC Crystal Input	
RTXO	O	RTC Crystal Output	

Pin Descriptions

Pin Name	I/O	Function	Driver Type
TCK	I	JTAG Clock	
TDO	O	JTAG Serial Data Out	C5
TDI	I	JTAG Serial Data In	
TMS	I	JTAG Mode Select	
TRST	I	JTAG Reset	
EMU	O	Emulation Output	C5
CLKIN	I	Clock/Crystal Input	
XTAL	O	Crystal Output	
RESET	I	Reset	
NMI	I	Nonmaskable Interrupt	

Pin Descriptions

Pin Name	I/O	Function	Driver Type
BMODE1–0	I	Boot Mode Strap	
VROUT1–0	O	External FET Drive	
VDDEXT	P	I/O Power Supply	
VDDINT	P	Core Power Supply	
VDDRTC	P	Real-Time Clock Power Supply	
GND	G	External Ground	

Introduction
Packaging

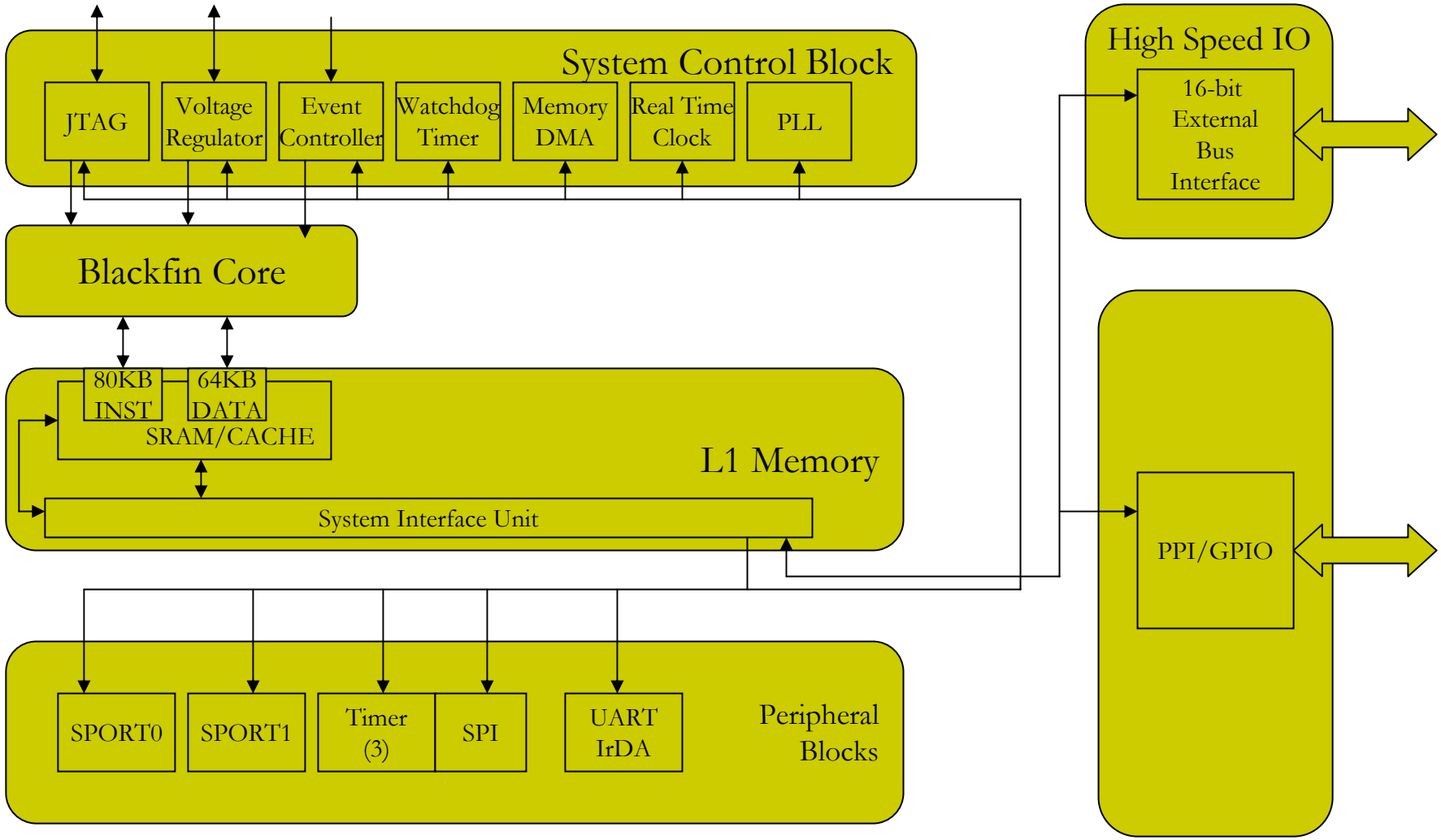
- Ball Grid Array (BGA) package
  - Improved electrical performance due to shorter distance between the chip and the solder balls
  - Improved thermal performance by use of thermal vias, or heat dissipation through power and ground planes incorporated in the substrate
  - Occupies less board real estate (less package area per I/O)
  - Significantly reduced handling-related lead damages due to use of solder balls instead of metal leads. This also reduces rework prior to board level attachment
  - When reflow attached to boards, the solder balls self align leading to higher manufacturing yields

Introduction
Packaging

- MiniBGA Package
  - 10x10 mm MiniBGA, approximately 50% smaller than most other DSPs
  - 144-ball chip array package (CAP) footprint, just one square centimeter
  - Height is just 1.25 millimeters

Architecture Overview

Single Core BlackFin 16/32 bit Processor



UIUC CS 433

Architecture Overview

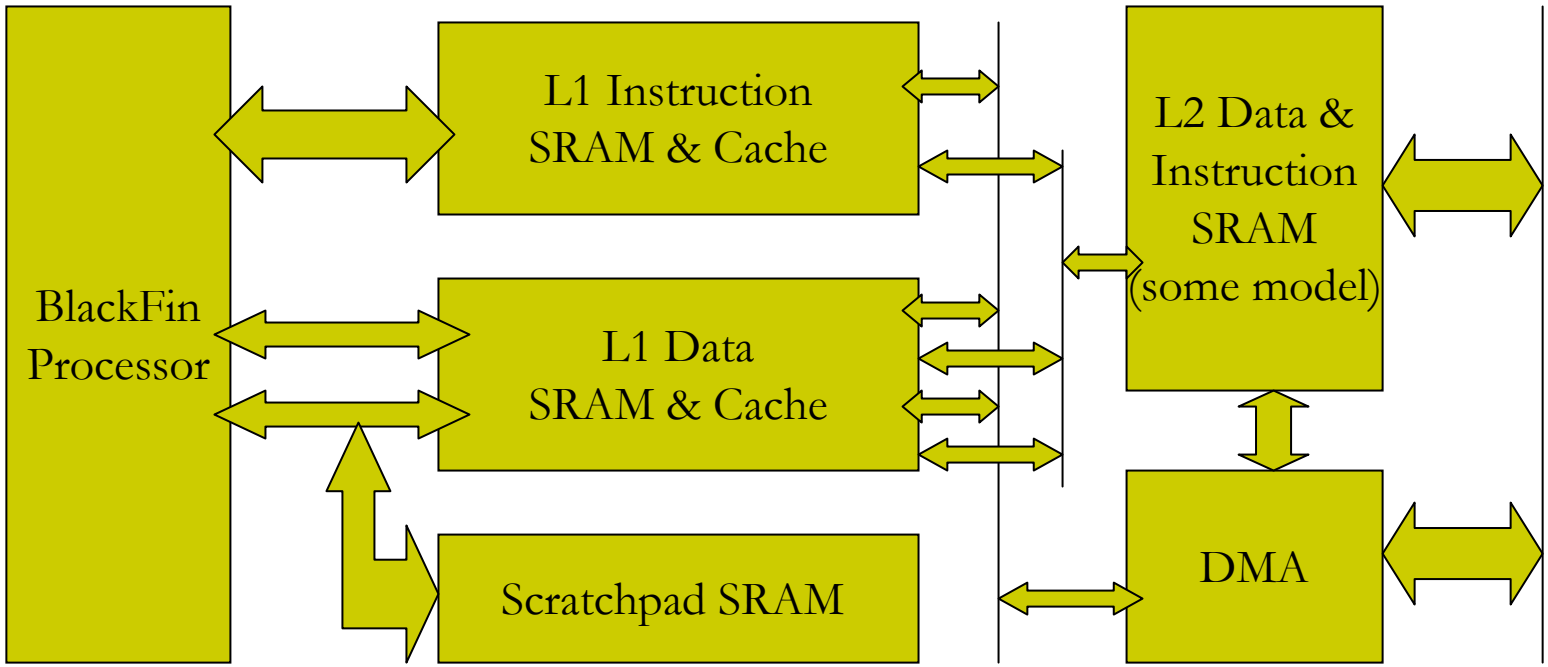
- **High Performance Processor Core**
  - 10-stage RISC MCU/DSP pipeline
  - mixed 16-/32-bit Instruction Set Architecture
  - fully SIMD compliant
  - instructions for accelerated video and image processing
  - full signal processing / analytical capabilities
  - efficient RISC MCU control tasking capabilities
- **High Bandwidth DMA Capability**
  - multiple, independent DMA controllers
  - automated data transfers with minimal overhead from the processor core

Architecture Overview

- **Video Instructions**
  - native support for 8-bit data
  - instructions specifically defined to enhance performance in video processing applications
  - allows OEMs to adapt to evolving standards and new functional requirements without hardware change
- **Efficient Control Processing**
  - powerful and flexible hierarchical memory architecture
  - superior code density
  - variety of microcontroller-style peripherals
  - great deal of design flexibility while minimizing end system costs

Architecture Overview

- **Hierarchical Memory**



Architecture Overview

- **Hierarchical Memory**
  - both Level 1 (L1) and Level 2 (L2) memory blocks
  - L1 connected directly to the processor core, runs at full system clock speed
  - L2 offers slightly reduced performance, but still faster than off-chip memory
  - L1 memory can be configured as SRAM, cache, or a combination of both
  - support a full Real Time Operating System with an isolated and secure environment for robust systems and applications

Architecture Overview

- **Addressing**
  - two address generation units that can each generate an independent address in each cycle
  - supports a variety of addressing modes, including: register indirect, register-indirect with post increment or post-decrement, register indirect indexed addressing with a short or long immediate offset, register-indirect addressing with pre-decrement for stack pushes, and register-indirect addressing with post-increment for stack pops.

Architecture Overview

- **Addressing**
  - can also perform some addition, subtraction, and shifting operations on the address registers
- **Pipelines**
  - first-generation ADSP-BF535 pipeline has eight stages
  - second-generation ADSP-BF5xx pipeline has ten stages
  - feature fully interlocked pipelines
  - static branch prediction for all conditional branches with programmer (or compiler) specify the prediction for each branch

Architecture Overview

- **Pipelines**
  - instructions generally execute in one cycle in the absence of memory access related delays and pipeline stalls
  - jump, call, and most return instructions require four or more cycles
- **Instruction Set**
  - algebraic syntax highly orthogonal
  - uses both 16- and 32-bit instructions
  - arithmetic instructions can take operands from the data registers, the pointer registers, the accumulators, or immediate operands
  - also support SIMD operations

Architecture Overview

- **Instruction Set**
  - supports multi-length instruction encoding
  - control-type instructions are encoded as compact 16-bit words
  - mathematically intensive signal processing instructions encoded as 32-bit values
  - intermix and link 16-bit control instructions with 32-bit signal processing instructions into 64-bit groups to maximize memory packing
  - code density comparable to industry-leading RISC processors.

Architecture Overview

- **Dynamic Power Management**
  - gated clock core design that selectively powers down functional units on an instruction-by-instruction basis
  - support multiple power-down modes for periods where little or no CPU activity is required
  - support a self contained dynamic power management scheme whereby the operating frequency AND voltage can be independently manipulated to meet the performance requirements of the algorithm currently being executed
- **Peripherals**
  - DMA controllers
  - general-purpose I/O pins
  - real-time clocks

Architecture Overview

- **Peripherals**
  - timers with pulse width modulation (PWM) and pulse measurement capability
  - watchdog timer
  - serial ports
  - UART ports
  - Serial Peripheral Interface (SPI) ports
- **Easy to Use**
  - utilized in many applications previously requiring both a high performance signal processor and a separate efficient control processor

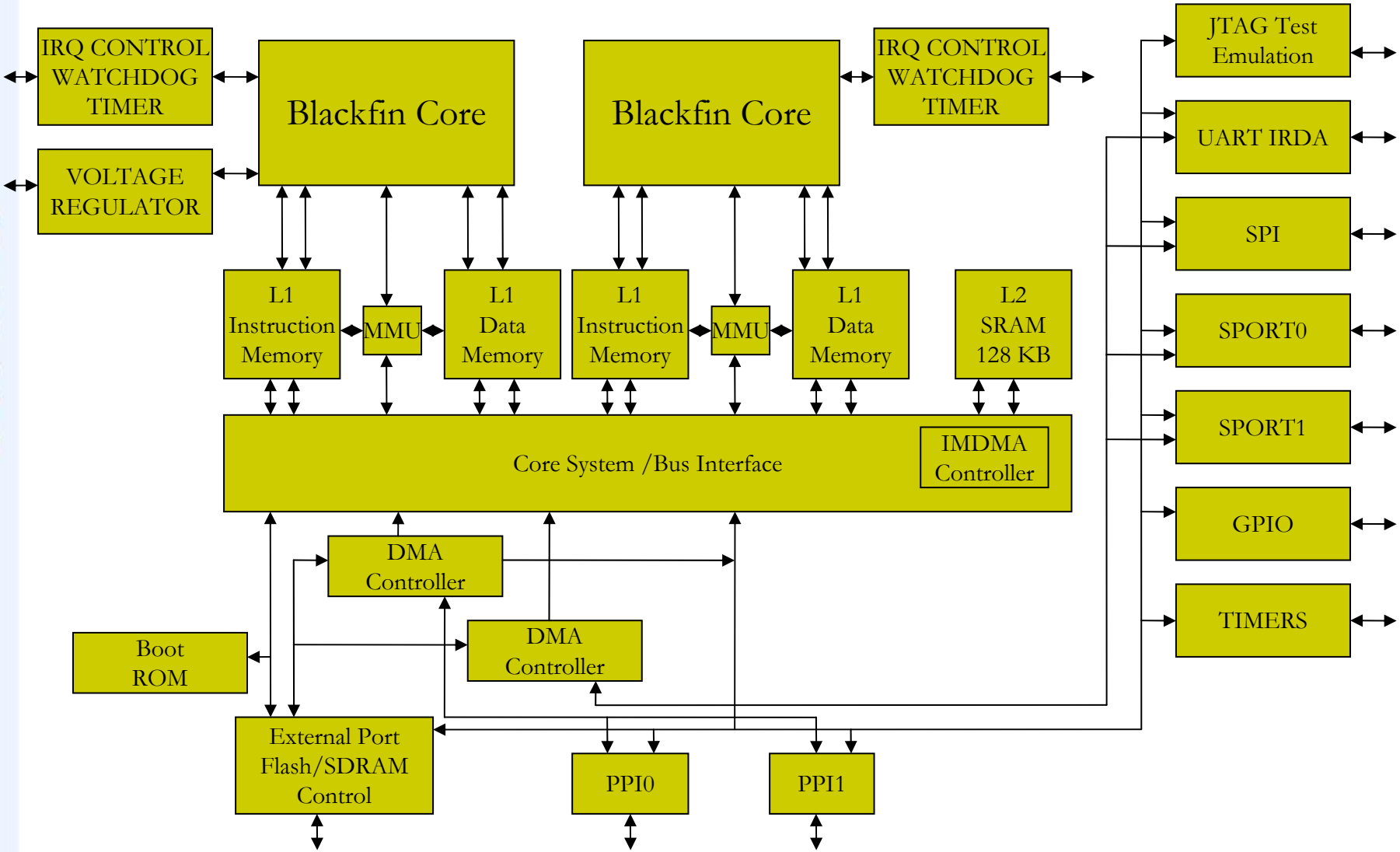
Architecture Overview

- **Major Benefits**
  - High-performance signal processing and efficient control processing capability enabling a variety of new markets and applications
  - Dynamic Power Management (DPM) enabling the system designer to specifically tailor the device power consumption profile to the end system requirements
  - Easy to use mixed 16-/32-bit Instruction Set Architecture and development tool suite ensuring that product development time is minimized
  - Low cost, priced starting at $4.95/each in 10K unit

Architecture Overview

Dual Core BlackFin 16/32 bit Processor

UIUC CS 433

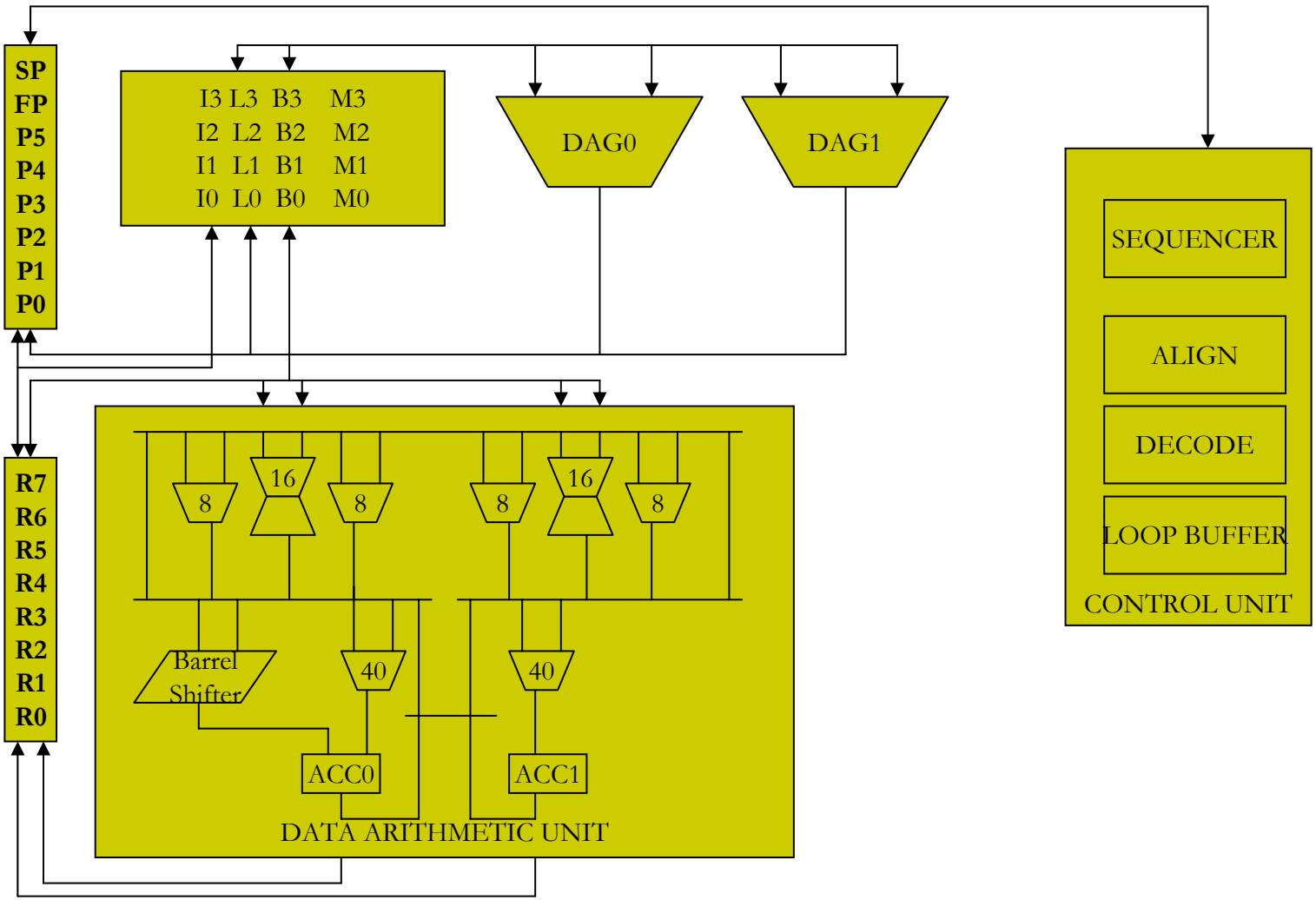


Architecture Overview

Dual Core BlackFin 16/32 bit Processor

- **Dual-Core Processors Add Flexibility**
  - employs discrete and often different tasks that run on each of the cores
  - one core runs the operating system or kernel, the other core is dedicated to the application's high-intensity processing functions
  - ability to segment these types of functions allows a parallel design process, eliminating critical path dependencies in the project

Processor Core



Processor Core

- General-purpose register files
  - Data register file
  - Data types include 8-, 16-, or 32-bit signed or unsigned integer and 16- or 32-bit signed fractional
  - 32-bit reads AND two 32-bit writes
  - Address register file
  - Stack pointer
  - Frame pointer

Processor Core

- Data arithmetic unit
  - Two 16-bit MACs
  - Two 40-bit ALUs
  - Four 8-bit video ALUs
  - Single barrel shifter

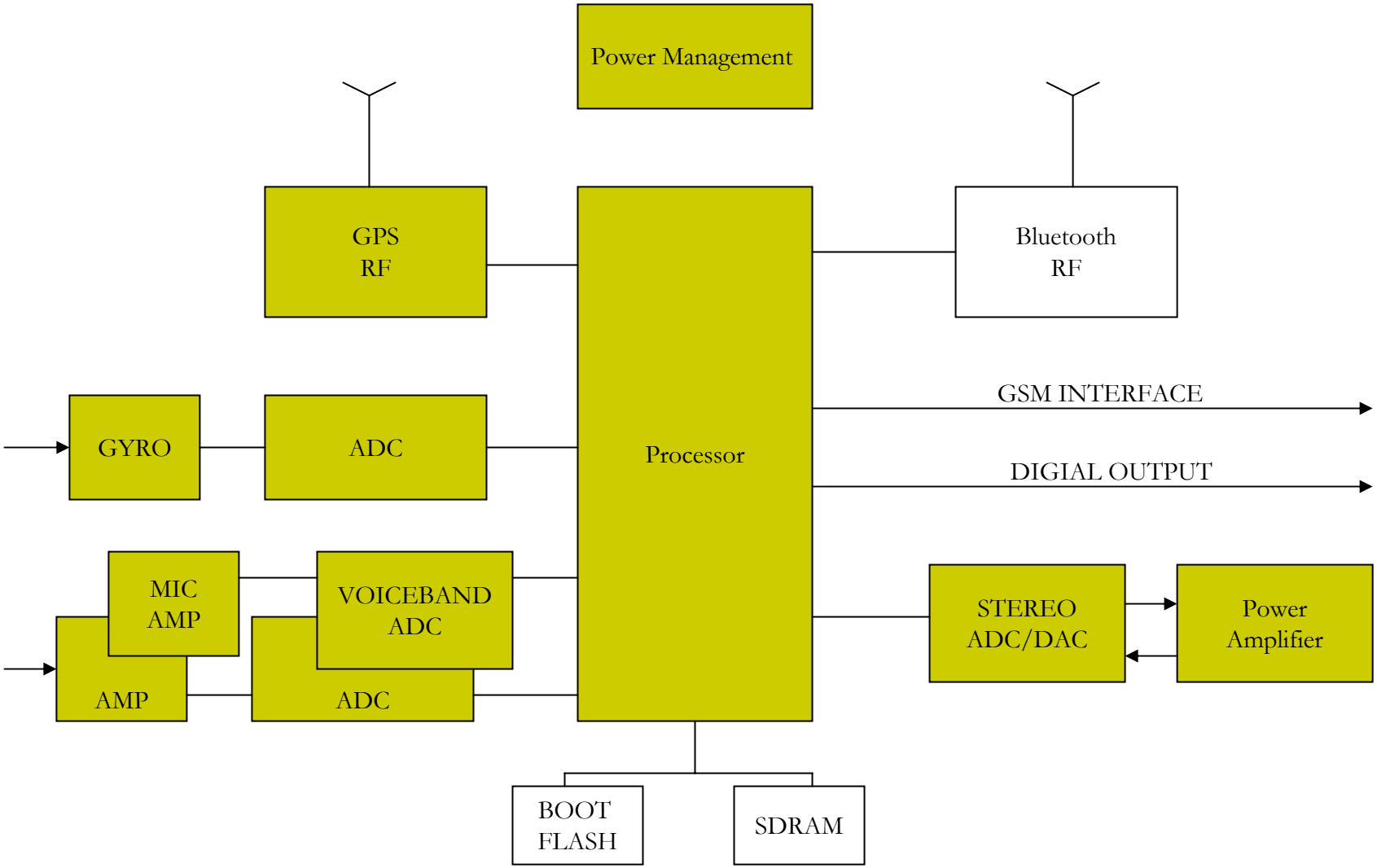
Processor Core

- Address arithmetic unit
  - Memory fetches
  - Index, length, base, and modify registers
  - Circular buffering

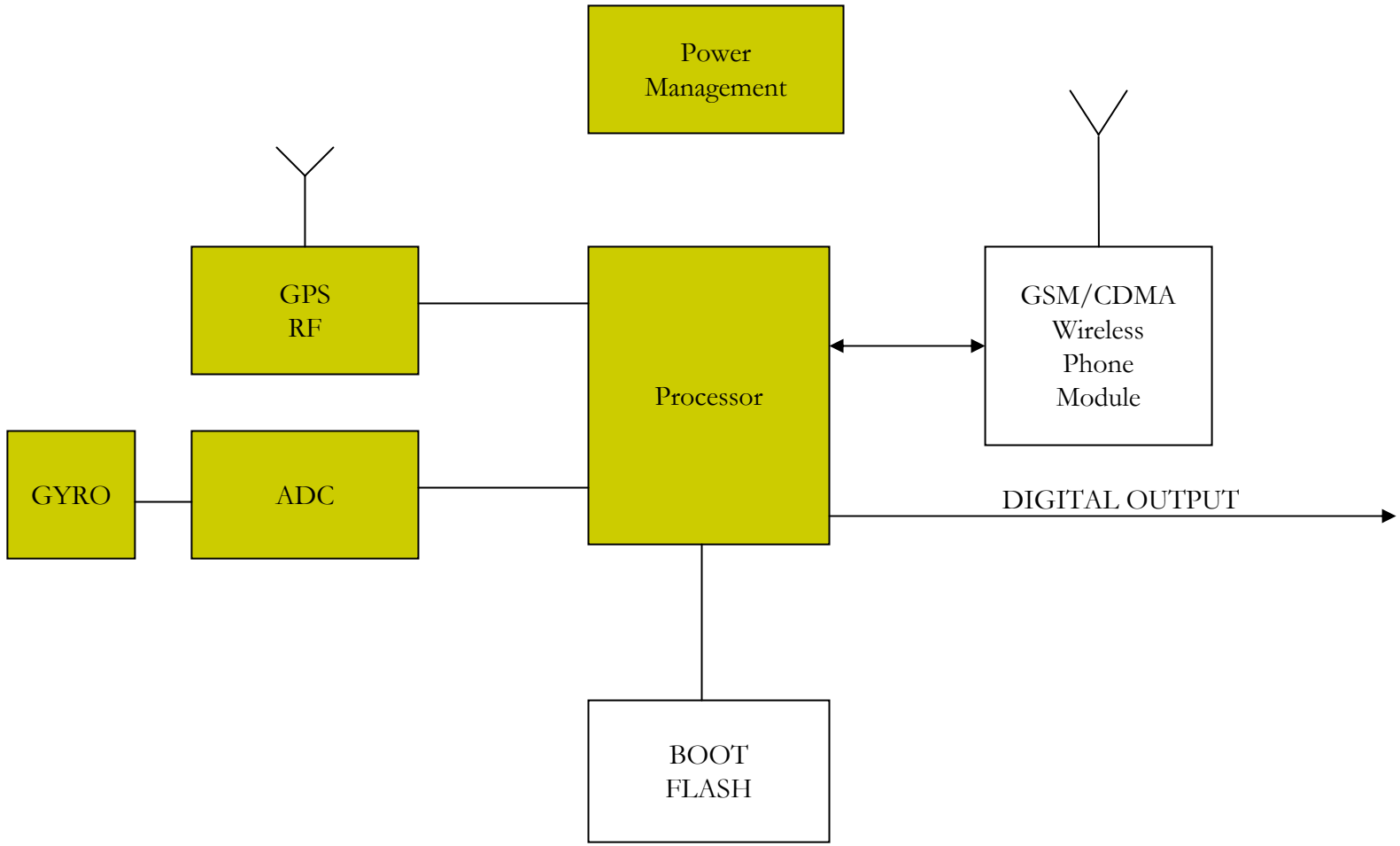
Processor Core

- Program sequencer unit
  - Conditional jumps and subroutine calls
  - Nested zero-overhead looping
  - Code density

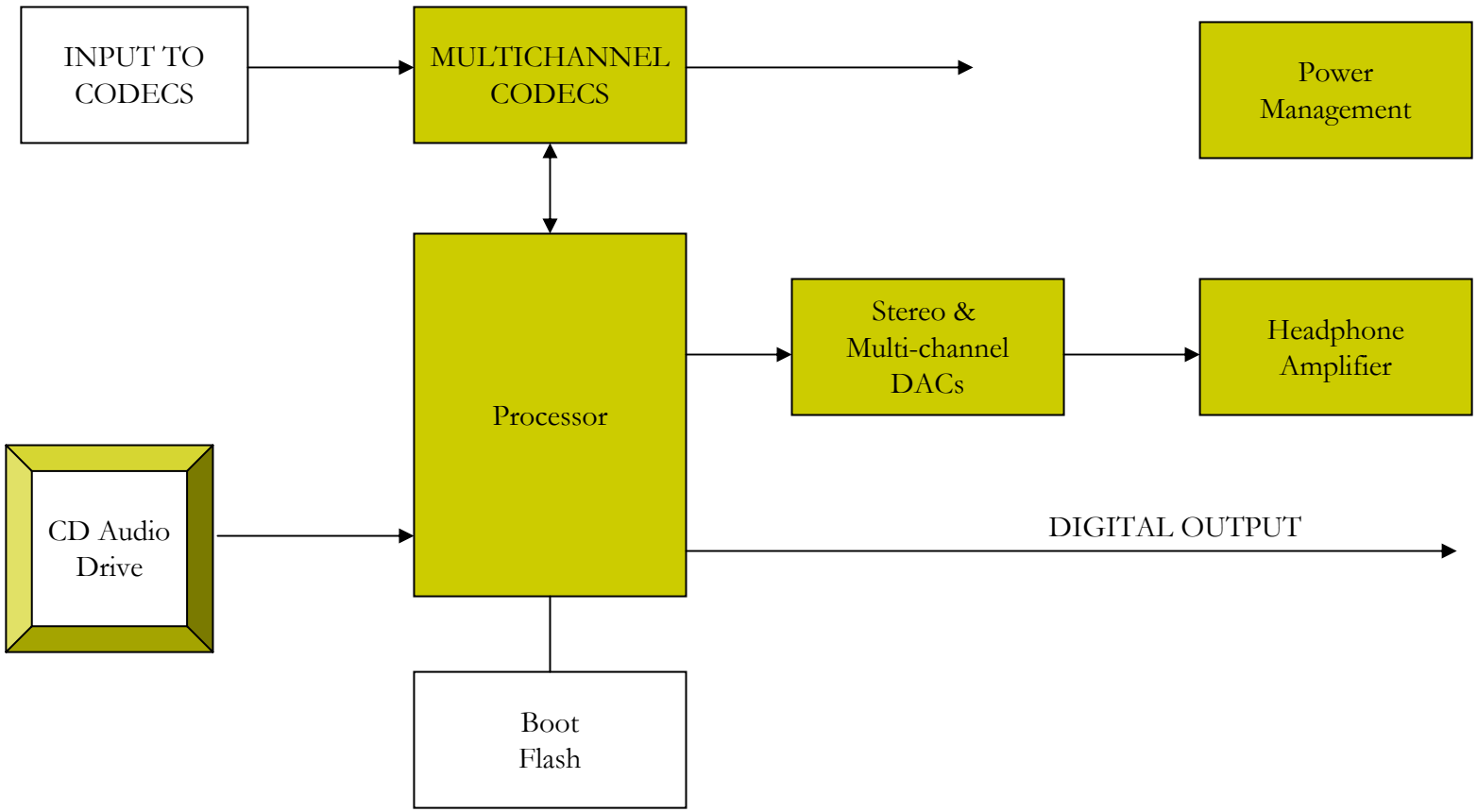
Signal Chain Telematics



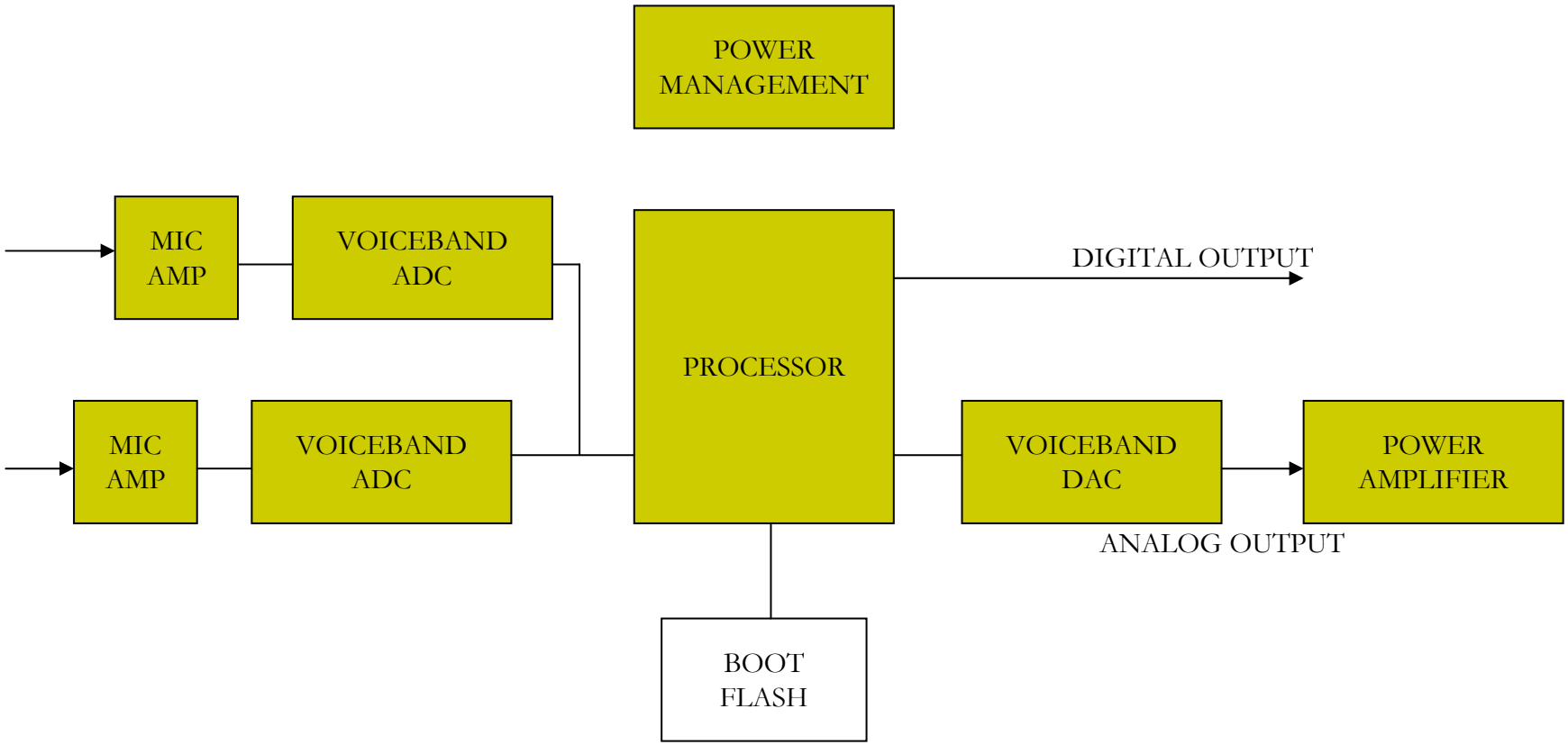
Signal Chain Navigation/GPS



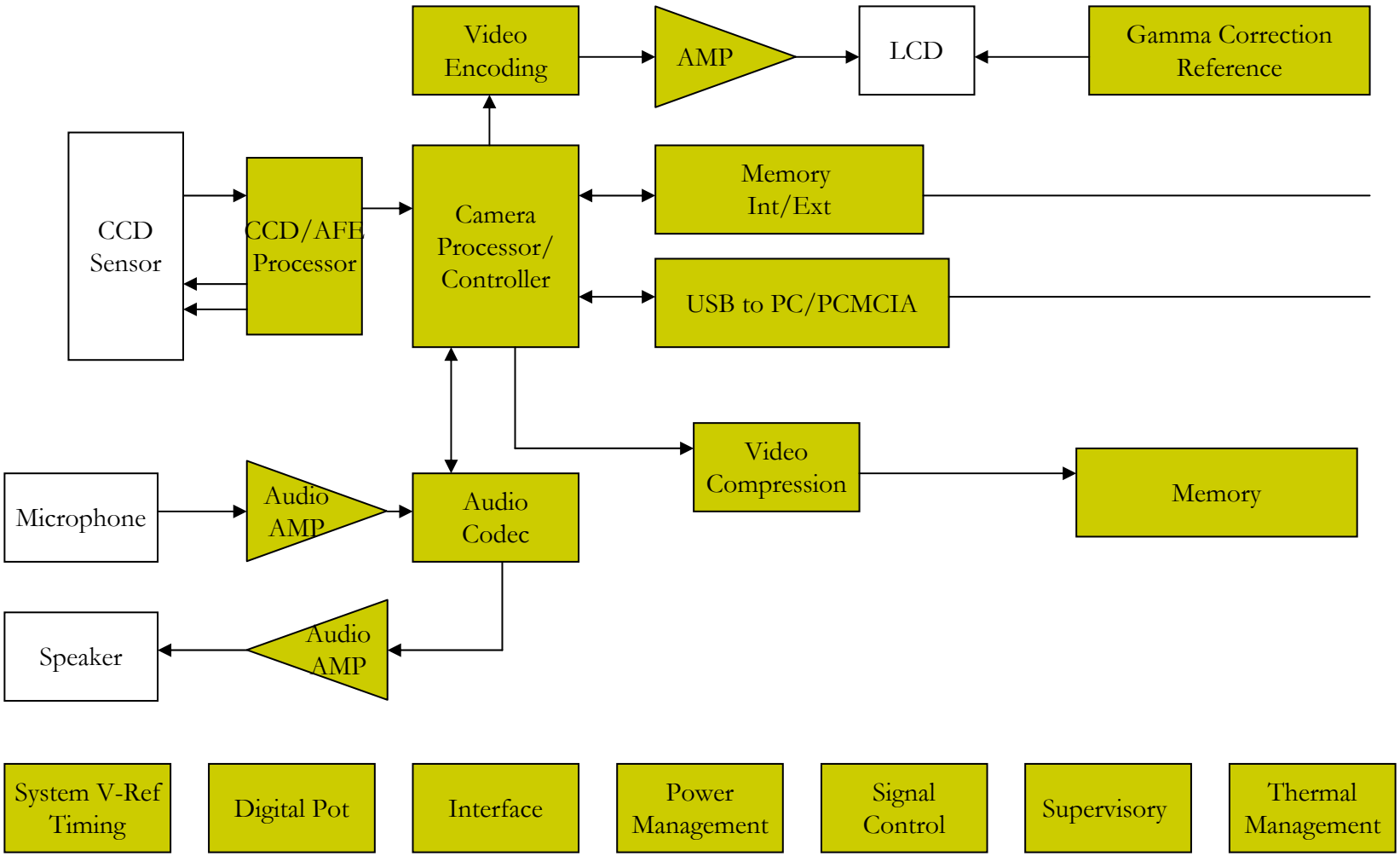
Signal Chain
Car Audio Amplifier



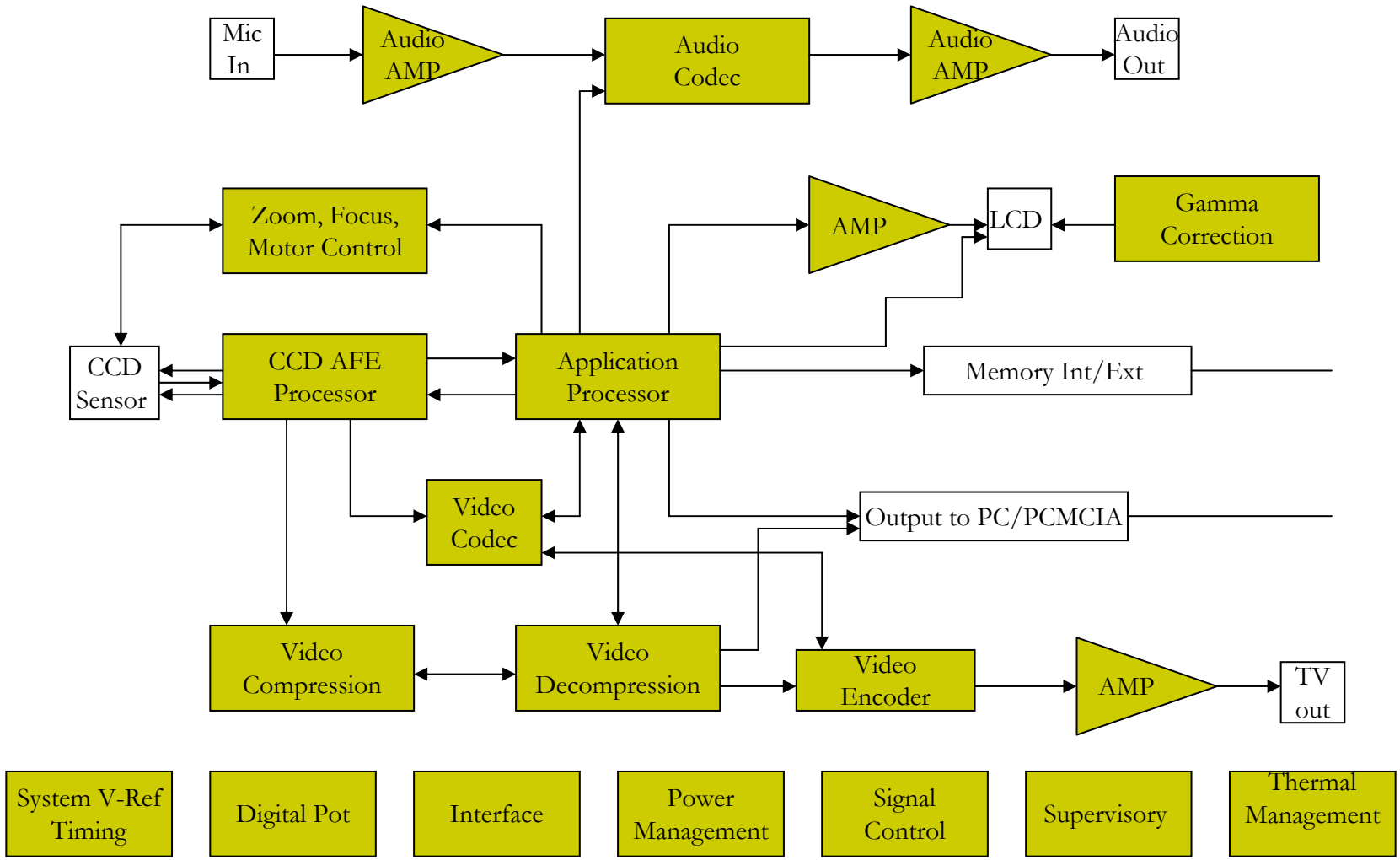
Hands Free/Voice Activated Control



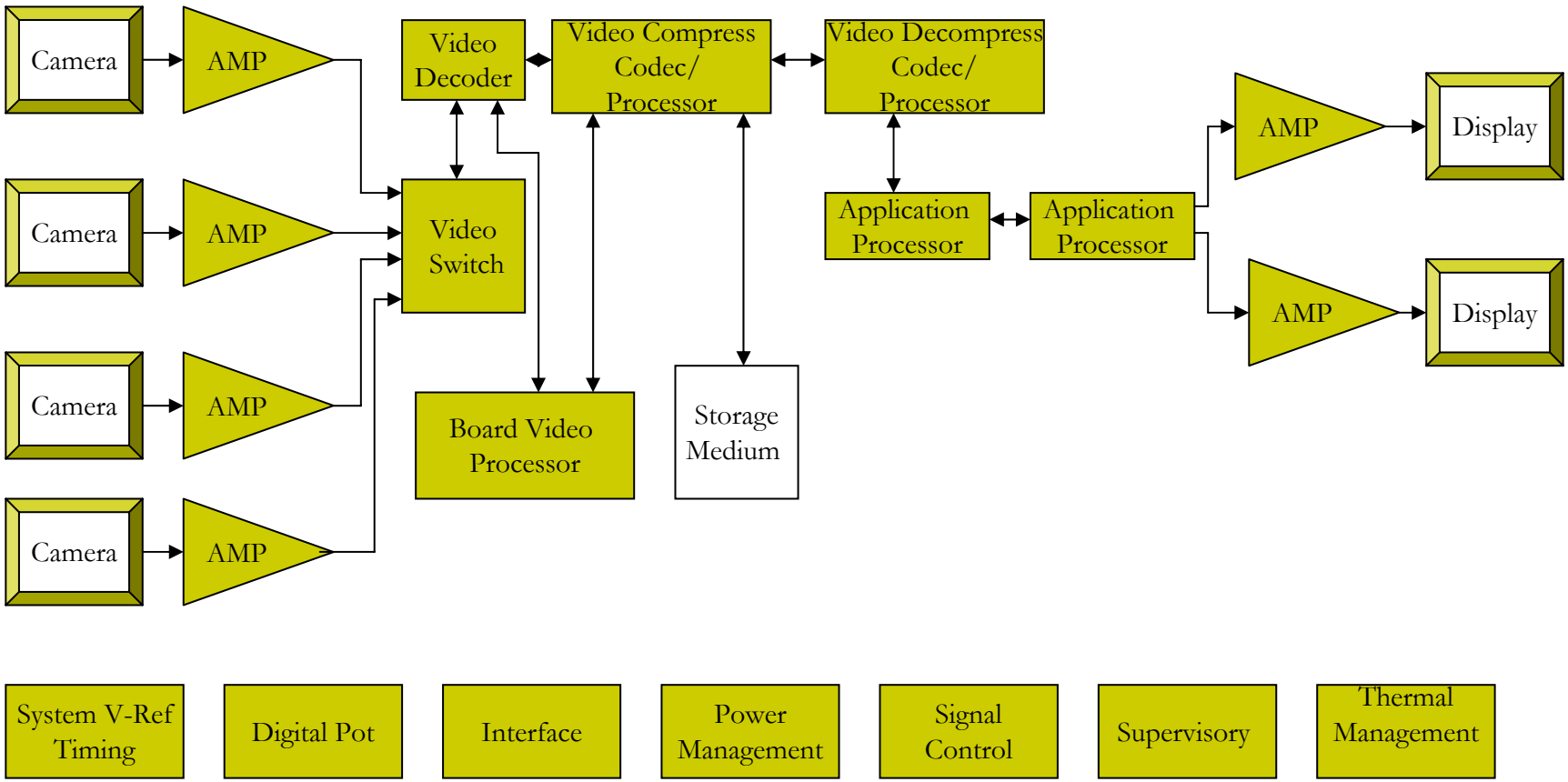
Signal Chain
Digital Camera



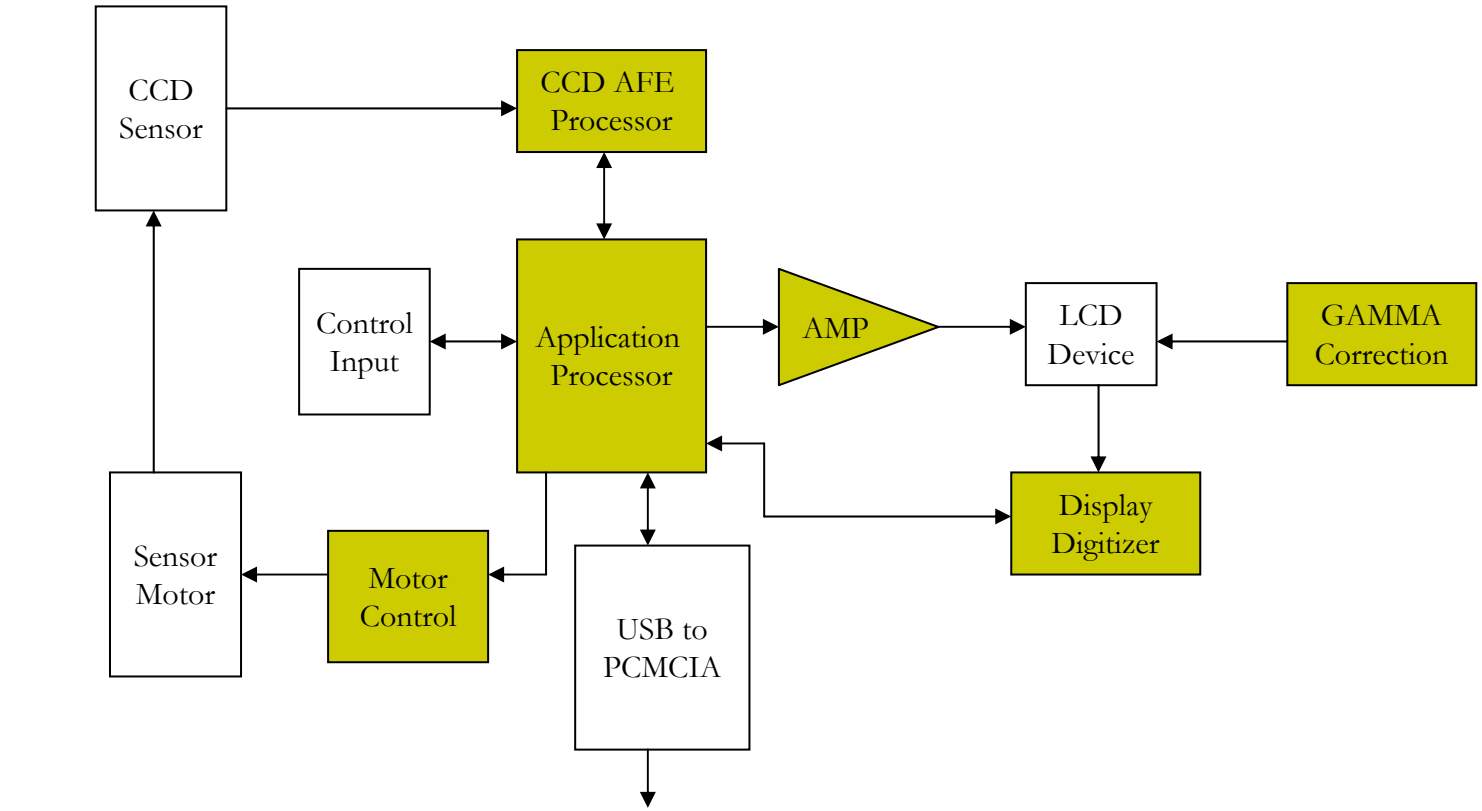
Signal Chain Camcorder



Signal Chain
Video Capture Board



Signal Chain
Image/Video – Document Scanner



System V-Ref Timing | Digital Pot | Interface | Power Management | Signal Control | Supervisory | Thermal Management

Instruction Set Description

- Seamlessly integrated DSP/CPU features are optimized for both 8-bit and 16-bit operations.
- A multi-issue load/store modified Harvard architecture, which supports two 16-bit MAC or four 8-bit ALU + two load/store + two pointer updates per cycle.
- All registers, I/O, and memory are mapped into a unified 4G byte memory space, providing a simplified programming model.

Instruction Set Description

- Microcontroller features, such as arbitrary bit and bit-field manipulation, insertion, and extraction; integer operations on 8-, 16-, and 32-bit data types; and separate user and supervisor stack pointers.
- Code density enhancements, which include intermixing of 16- and 32-bit instructions (no mode switching, no code segregation). Frequently used instructions are encoded in 16 bits.

Program Flow Control

- Program Flow Control
  - Jump
    - JUMP (destination_indirect)
    - JUMP (PC + offset)
    - JUMP offset
    - JUMP.S offset
    - JUMP.L offset
  - IF CC JUMP
    - IF CC JUMP destination
    - IF !CC JUMP destination

Program Flow Control

- Program Flow Control
  - Call
    - CALL (destination_indirect)
    - CALL (PC + offset)
    - CALL offset
  - RTS, RTI, RTX, RTN, RTE (Return)
    - RTS ; // Return from Subroutine (a)
    - RTI ; // Return from Interrupt (a)
    - RTX ; // Return from Exception (a)
    - RTN ; // Return from NMI (a)
    - RTE ; // Return from Emulation (a)

Program Flow Control

- Program Flow Control
  - LSETUP, LOOP
    - There are two forms of this instruction. The first is:
      - LOOP loop_name loop_counter
      - LOOP_BEGIN loop_name
      - LOOP_END loop_name
    - The second form is:
      - LSETUP (Begin_Loop, End_Loop)Loop_Counter

Load / Store

- Load / Store
  - Load Immediate
    - register = constant
    - A1 = A0 = 0
  - Load Pointer Register
    - P-register = [indirect_address]
  - Load Data Register
    - D-register = [indirect_address]
  - Load Half-Word – Zero-Extended
    - D-register = W [indirect_address] (Z)

Load / Store

- Load / Store
  - Load Half-Word – Sign-Extended
    - D-register = W [indirect_address] (X)
  - Load High Data Register Half
    - Dreg_hi = W [indirect_address]
  - Load Low Data Register Half
    - Dreg_lo = W [indirect_address]
  - Load Low Data Register Half
    - Dreg_lo = W [indirect_address]

Load / Store

- Load / Store
  - Load Byte – Sign-Extended
    - D-register = B [indirect_address] (X)
  - Store Pointer Register
    - [indirect_address] = P-register
  - Store Data Register
    - [indirect_address] = D-register
  - Store High Data Register Half
    - W [indirect_address] = Dreg_hi

Load / Store

- Load / Store
  - Store Low Data Register Half
    - W [indirect_address] = Dreg_lo
    - W [indirect_address] = D-register
  - Store Byte
    - B [indirect_address] = D-register
- Move
  - Move Register
    - dest_reg = src_reg

Instruction Set Move

- Move
  - Move Conditional
    - IF CC dest_reg = src_reg
    - IF ! CC dest_reg = src_reg
  - Move Half to Full Word – Zero-Extended
    - dest_reg = src_reg (Z)
  - Move Half to Full Word – Sign-Extended
    - dest_reg = src_reg (X)

Instruction Set Move

- Move
  - Move Register Half
    - dest_reg_half = src_reg_half
    - dest_reg_half = accumulator (opt_mode)
  - Move Byte – Zero-Extended
    - dest_reg = src_reg_byte (Z)
  - Move Byte – Sign-Extended
    - dest_reg = src_reg_byte (X)

Instruction Set
Stack Control

- Stack Control
  - --SP (Push)
    - [-- SP] = src_reg
  - --SP (Push Multiple)
    - [-- SP] = (src_reg_range)
  - SP++ (Pop)
    - dest_reg = [SP ++]
  - SP++ (Pop Multiple)
    - (dest_reg_range) = [SP ++]

Instruction Set
Control Code Bit Management

- Stack Control
  - LINK, UNLINK
    - LINK *uimm18m4* ; /* allocate a stack frame of specified size (b) */
    - UNLINK ; /* de-allocate the stack frame (b)*/
- Control Code Bit Management
  - Compare Data Register
    - CC = operand_1 == operand_2
    - CC = operand_1 < operand_2
    - CC = operand_1 <= operand_2
    - CC = operand_1 < operand_2 (IU)
    - CC = operand_1 <= operand_2 (IU)

Instruction Set
Control Code Bit Management

- Control Code Bit Management
  - Compare Pointer
    - CC = operand_1 == operand_2
    - CC = operand_1 < operand_2
    - CC = operand_1 <= operand_2
    - CC = operand_1 < operand_2 (IU)
    - CC = operand_1 <= operand_2 (IU)
  - Compare Accumulator
    - CC = A0 == A1
    - CC = A0 < A1
    - CC = A0 <= A1

Instruction Set
Control Code Bit Management

- Control Code Bit Management
  - Move CC
    - dest = CC
    - dest |= CC
    - dest &= CC
    - dest ^= CC
    - CC = source
    - CC |= source
    - CC &= source
    - CC ^= source

Instruction Set Logical Operations

- Control Code Bit Management
  - Negate CC
    - CC = ! CC
- Logical Operations
  - & (AND)
    - dest_reg = src_reg_0 & src_reg_1
  - ~ (NOT One's Complement)
    - dest_reg = ~ src_reg

Instruction Set Logical Operations

- Logical Operations
  - | (OR)
    - dest_reg = src_reg_0 | src_reg_1
  - ^ (Exclusive-OR)
    - dest_reg = src_reg_0 ^ src_reg_1
  - BXORSHIFT, BXOR
    - dest_reg = CC = BXORSHIFT (A0, src_reg)
    - dest_reg = CC = BXOR (A0, src_reg)
    - dest_reg = CC = BXOR (A0, A1, CC)
    - A0 = BXORSHIFT (A0, A1, CC)

Instruction Set
Bit Operations

- Bit Operations
  - BITCLR
    - BITCLR (register, bit_position)
  - BITSET
    - BITSET (register, bit_position)
  - BITTGL
    - BITTGL (register, bit_position)
  - BITTST
    - CC = BITTST (register, bit_position)
    - CC = ! BITTST (register, bit_position)

Instruction Set
Bit Operations

- Bit Operations
  - DEPOSIT
    - dest_reg = DEPOSIT (backgnd_reg, foregnd_reg)
    - dest_reg = DEPOSIT (backgnd_reg, foregnd_reg) (X)
  - EXTRACT
    - dest_reg = EXTRACT (scene_reg, pattern_reg) (Z)
    - dest_reg = EXTRACT (scene_reg, pattern_reg) (X)
  - BITMUX
    - BITMUX (source_1, source_0, A0) (ASR)
    - BITMUX (source_1, source_0, A0) (ASL)

Instruction Set
Shift / Rotate Operations

- Bit Operatoins
  - ONES (One's Population Count)
    - dest_reg = ONES src_reg
- Shift / Rotate Operations
  - Add with Shift
    - dest_pntr = (dest_pntr + src_reg) << 1
    - dest_pntr = (dest_pntr + src_reg) << 2
    - dest_reg = (dest_reg + src_reg) << 1
    - dest_reg = (dest_reg + src_reg) << 2

Instruction Set
Shift / Rotate Operations

- Shift / Rotate Operations
  - Shift with Add
    - dest_pntr = adder_pntr + (src_pntr << 1)
    - dest_pntr = adder_pntr + (src_pntr << 2)
  - Arithmetic Shift
    - dest_reg >>>= shift_magnitude
    - dest_reg = src_reg >>> shift_magnitude (opt_sat)
    - dest_reg = src_reg << shift_magnitude (S)
    - accumulator = accumulator >>> shift_magnitude
    - dest_reg = ASHIFT src_reg BY shift_magnitude (opt_sat)
    - accumulator = ASHIFT accumulator BY shift_magnitude

Instruction Set
Shift / Rotate Operations

- Shift / Rotate Operations
  - Logical Shift
    - dest_pntr = src_pntr >> 1
    - dest_pntr = src_pntr >> 2dest_pntr = src_pntr << 1
    - dest_pntr = src_pntr << 2dest_reg >>= shift_magnitude
    - dest_reg <<= shift_magnitude
    - dest_reg = src_reg >> shift_magnitude
    - dest_reg = src_reg << shift_magnitude
    - dest_reg = LSHIFT src_reg BY shift_magnitude

Instruction Set Arithmetic Operations

- Shift / Rotate Operations
  - ROT (Rotate)
    - dest_reg = ROT src_reg BY rotate_magnitude
    - accumulator_new = ROT accumulator_old BY rotate_magnitude
- Arithmetic Operations
  - ABS
    - dest_reg = ABS src_reg
  - Add
    - dest_reg = src_reg_1 + src_reg_2

Instruction Set Arithmetic Operations

- Arithmetic Operations
  - Add/Subtract – Prescale Down
    - dest_reg = src_reg_0 + src_reg_1 (RND20)
    - dest_reg = src_reg_0 - src_reg_1 (RND20)
  - Add/Subtract – Prescale Up
    - dest_reg = src_reg_0 + src_reg_1 (RND12)
    - dest_reg = src_reg_0 - src_reg_1 (RND12)
  - Add Immediate
    - register += constant

Instruction Set Arithmetic Operations

- Arithmetic Operations
  - DIVS, DIVQ (Divide Primitive)
    - DIVS (dividend_register, divisor_register)
    - DIVQ (dividend_register, divisor_register)
  - EXPADJ
    - dest_reg = EXPADJ (sample_register, exponent_register)
  - MAX
    - dest_reg = MAX (src_reg_0, src_reg_1)
  - MIN
    - dest_reg = MIN (src_reg_0, src_reg_1)

Instruction Set
Arithmetic Operations

- Arithmetic Operations
  - Modify – Decrement
    - dest_reg -= src_reg
  - Modify – Increment
    - dest_reg += src_reg
    - dest_reg = (src_reg_0 += src_reg_1)
  - Multiply 16-Bit Operands
    - dest_reg = src_reg_0 * src_reg_1 (opt_mode)
  - Multiply 32-Bit Operands
    - dest_reg *= multiplier_register

Instruction Set Arithmetic Operations

- Arithmetic Operations
  - Multiply and Multiply-Accumulate to Accumulator
    - accumulator = src_reg_0 * src_reg_1 (opt_mode)
    - accumulator += src_reg_0 * src_reg_1 (opt_mode)
    - accumulator –= src_reg_0 * src_reg_1 (opt_mode)
  - Multiply and Multiply-Accumulate to Half-Register
    - dest_reg_half = (accumulator = src_reg_0 * src_reg_1) (opt_mode)
    - dest_reg_half = (accumulator += src_reg_0 * src_reg_1) (opt_mode)
    - dest_reg_half = (accumulator –= src_reg_0 * src_reg_1) (opt_mode)

Instruction Set
Arithmetic Operations

- Arithmetic Operations
  - Multiply and Multiply-Accumulate to Data Register
    - dest_reg = (accumulator = src_reg_0 * src_reg_1) (opt_mode)
    - dest_reg = (accumulator += src_reg_0 * src_reg_1) (opt_mode)
    - dest_reg = (accumulator –= src_reg_0 * src_reg_1) (opt_mode)
  - Negate (Two's Complement)
    - dest_reg = – src_reg
    - dest_accumulator = – src_accumulator
  - RND (Round to Half-Word)
    - dest_reg = src_reg (RND)

Instruction Set Arithmetic Operations

- Arithmetic Operations
  - Saturate
    - dest_reg = src_reg (S)
  - SIGNBITS
    - dest_reg = SIGNBITS sample_register
  - Subtract
    - dest_reg = src_reg_1 - src_reg_2
  - Subtract Immediate
    - register -= constant

Instruction Set
External Event Management

- External Event Management
  - Idle
    - IDLE
  - Core Synchronize
    - CSYNC
  - System Synchronize
    - SSYNC
  - EMUEXCPT (Force Emulation)
    - EMUEXCPT

Instruction Set
External Event Management

- External Event Management
  - Disable Interrupts
    - CLI
  - Enable Interrupts
    - STI
  - RAISE (Force Interrupt / Reset)
    - RAISE
  - EXCPT (Force Exception)
    - EXCPT

Instruction Set Cache Control

- External Event Management
  - Test and Set Byte (Atomic)
    - TESTSET
  - No Op
    - NOP
    - MNOP
- Cache Control
  - PREFETCH
    - PREFETCH [*Preg*] ; /* indexed (a) */
    - PREFETCH [*Preg* ++] ; /* indexed, post increment (a) */

Instruction Set Cache Control

- Cache Control
  - FLUSH
    - FLUSH [*Preg*] ; /* indexed (a) */
    - FLUSH [*Preg* ++] ; /* indexed, post increment (a) */
  - FLUSHINV
    - FLUSHINV [*Preg*] ; /* indexed (a) */
    - FLUSHINV [*Preg* ++] ; /* indexed, post increment (a) */
  - IFLUSH
    - IFLUSH [*Preg*] ; /* indexed (a) */
    - IFLUSH [*Preg* ++] ; /* indexed, post increment (a) */

Instruction Set
Video Pixel Operations

- Video Pixel Operations
  - ALIGN8, ALIGN16, ALIGN24
    - dest_reg = ALIGN8 (src_reg_1, src_reg_0)
    - dest_reg = ALIGN16 (src_reg_1, src_reg_0)
    - dest_reg = ALIGN24 (src_reg_1, src_reg_0)
  - DISALGNEXCPT
    - DISALGNEXCPT
  - BYTEOP3P (Dual 16-Bit Add / Clip)
    - dest_reg = BYTEOP3P (src_reg_0, src_reg_1) (LO)
    - dest_reg = BYTEOP3P (src_reg_0, src_reg_1) (HI)

Instruction Set
Video Pixel Operations

- Video Pixel Operation
  - BYTEOP3P (Dual 16-Bit Add / Clip)
    - dest_reg = BYTEOP3P (src_reg_0, src_reg_1) (LO, R)
    - dest_reg = BYTEOP3P (src_reg_0, src_reg_1) (HI, R)
  - Dual 16-Bit Accumulator Extraction with Addition
    - dest_reg_1 = A1.L + A1.H, dest_reg_0 = A0.L + A0.H
  - BYTEOP16P (Quad 8-Bit Add)
    - (dest_reg_1, dest_reg_0) = BYTEOP16P (src_reg_0, src_reg_1)
    - (dest_reg_1, dest_reg_0) = BYTEOP16P (src_reg_0, src_reg_1) (R)

Instruction Set
Video Pixel Operations

- Video Pixel Operation
  - BYTEOP1P (Quad 8-Bit Average – Byte)
    - dest_reg = BYTEOP1P (src_reg_0, src_reg_1)
    - dest_reg = BYTEOP1P (src_reg_0, src_reg_1) (T)
    - dest_reg = BYTEOP1P (src_reg_0, src_reg_1) (R)
    - dest_reg = BYTEOP1P (src_reg_0, src_reg_1) (T, R)
  - BYTEOP2P (Quad 8-Bit Average – Half-Word)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (RNDL)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (RNDH)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (TL)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (TH)

Instruction Set
Video Pixel Operations

- Video Pixel Operations
  - BYTEOP2P (Quad 8-Bit Average – Half-Word)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (RNDL, R)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (RNDH, R)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (TL, R)
    - dest_reg = BYTEOP2P (src_reg_0, src_reg_1) (TH, R)
  - BYTEPACK (Quad 8-Bit Pack)
    - dest_reg = BYTEPACK (src_reg_0, src_reg_1)
  - BYTEOP16M (Quad 8-Bit Subtract)
    - (dest_reg_1, dest_reg_0) = BYTEOP16M (src_reg_0, src_reg_1)
    - (dest_reg_1, dest_reg_0) = BYTEOP16M (src_reg_0, src_reg_1) (R)

Instruction Set
Vector Operations

- Video Pixel Operations
  - SAA (Quad 8-Bit Subtract-Absolute-Accumulate)
    - SAA (src_reg_0, src_reg_1)
    - SAA (src_reg_0, src_reg_1) (R)
  - BYTEUNPACK (Quad 8-Bit Unpack)
    - (dest_reg_1, dest_reg_0) = BYTEUNPACK src_reg_pair
    - (dest_reg_1, dest_reg_0) = BYTEUNPACK src_reg_pair (R)
- Vector Operations
  - Add on Sign
    - dest_hi = dest_lo = SIGN (src0_hi) * src1_hi + SIGN (src0_lo) * src1_lo

Instruction Set
Vector Operations

- Vector Operations
  - VIT_MAX (Compare-Select)
    - dest_reg = VIT_MAX (src_reg_0, src_reg_1) (ASL)
    - dest_reg = VIT_MAX (src_reg_0, src_reg_1) (ASR)
    - dest_reg_lo = VIT_MAX (src_reg) (ASL)
    - dest_reg_lo = VIT_MAX (src_reg) (ASR)
  - Vector ABS
    - dest_reg = ABS source_reg (V)
  - Vector Add / Subtract
    - dest = src_reg_0 +|+ src_reg_1

Instruction Set Vector Operations

- Vector Operations
  - Vector Add / Subtract
    - dest = src_reg_0 –|+ src_reg_1
    - dest = src_reg_0 +|– src_reg_1
    - dest = src_reg_0 –|– src_reg_1
    - dest_0 = src_reg_0 +|+ src_reg_1,
    - dest_1 = src_reg_0 –|– src_reg_1
    - dest_0 = src_reg_0 +|– src_reg_1,
    - dest_1 = src_reg_0 –|+ src_reg_1
    - dest_0 = src_reg_0 + src_reg_1,
    - dest_1 = src_reg_0 – src_reg_1
    - dest_0 = A1 + A0, dest_1 = A1 – A0
    - dest_0 = A0 + A1, dest_1 = A0 – A1

Instruction Set
Vector Operations

- Vector Operations
  - Vector Arithmetic Shift
    - dest_reg = src_reg >>> shift_magnitude (V)
    - dest_reg = ASHIFT src_reg BY shift_magnitude (V)
  - Vector Logical Shift
    - dest_reg = src_reg >> shift_magnitude (V)
    - dest_reg = src_reg << shift_magnitude (V)
    - dest_reg = LSHIFT src_reg BY shift_magnitude (V)
  - Vector MAX
    - dest_reg = MAX (src_reg_0, src_reg_1) (V)

Instruction Set
Vector Operations

- Vector Operations
  - Vector MIN
    - dest_reg = MIN (src_reg_0, src_reg_1) (V)
  - Vector Negate (Two's Complement)
    - dest_reg = – source_reg (V)
  - Vector PACK
    - Dest_reg = PACK (src_half_0, src_half_1)
  - Vector SEARCH
    - (dest_pointer_hi, dest_pointer_lo) = SEARCH src_reg (searchmode)

Instruction Set
Parallel Issue Instructions

- Parallel Issue Instructions
  - Supported Parallel Combinations
    - *A 32-bit ALU/MAC instruction* || *A 16-bit instruction* || *A 16-bit instruction* ;
    - *A 32-bit ALU/MAC instruction* || *A 16-bit instruction* ;
    - MNOP || *A 16-bit instruction* || *A 16-bit instruction* ;
  - 32-Bit ALU/MAC Instructions
  - 16-Bit Instructions
    - The two 16-bit instructions in a multi-issue instruction must each be from Group1 and Group2 instructions
    - Only one of the 16-bit instructions can be a store instruction
    - If the two 16-bit instructions are memory access instructions, then
    - both cannot use P-registers as address registers. In this case, at least
    - one memory access instruction must be an I-register version.

Instruction Set
Parallel Issue Instructions

- Parallel Issue Instructions
  - Examples
    - Two Parallel Memory Access Instructions
      - saa (r1:0, r3:2) || r0=[i0++] || r2=[i1++] ;
    - One *Ireg* and One Memory Access Instruction in Parallel
      - r7.h=r7.l=sign(r2.h)*r3.h + sign(r2.l)*r3.l || i0+=m3 || r0=[i0];
    - One *Ireg* Instruction in Parallel
      - r6=(a0+=r3.h*r2.h)(fu) || i2-=m0 ;

Blackfin Platforms

- **Blackfin eMedia Platform**
  - ideally suited for IP set-top boxes, media servers, portable entertainment devices, DTVs, residential data/media gateways and networked digital media adapters
  - Multiple digital video formats - Windows Media 9 Series / VC-1, MPEG-2, MPEG-4, H264 AVC and future format
  - Multiple audio formats including WMA, MP3, MP3 PRO, AAC, and others
  - G.711, G.728, G.729AB, G.723.1A, AMR for Speech
  - On-screen user interfaces and displays

Blackfin Platforms

- **Blackfin eMedia Platform**
  - A wide range of home-network protocols, such as UPnP, TCP/IP, Ethernet, HPNA 2.0, 802.11a/b/g and HomePlug
  - Access and management of centralized or distributed media
  - Remote firmware upgrade to enable new formats and technologies as they are introduced
- **Blackfin Car Telematics Platform**
  - serve the embedded processing market for feature-rich, multimedia car telematics applications

Blackfin Platforms

- **Blackfin Car Telematics Platform**
  - meets the computational demands and power constraints of in-vehicle safety systems, audio, video, and wireless communications
  - functions implemented on the Blackfin Processor include GPS, hands free operation, microphone array, voice-activated control, GSM interfaces, and car audio play/record of MP3 and WMA content
  - fully supported with speech recognition and text-to-speech algorithms available from Scansoft, Inc. and with the noise- and echo-cancellation algorithms from Bitwave and Clarity

Blackfin Platforms

- **Blackfin Car Telematics Platform**
  - includes a collection of telematics function modules that reduce code-development time and enable faster time-to-market
- **Blackfin BRAVO Platform**
  - deliver audio, streaming video, networked camera and videophone capabilities with up to full D1, MPEG4/MPEG2/DivX/WMV9 30 fps full color, full motion video in CIF resolution over broadband networks

Blackfin Platforms

- **Blackfin BRAVO Platform**
  - include an Ethernet port to interface with cable, xDSL, Ethernet, USB, IEEE 802.11x and fiber, and support Microsoft Windows Media, ISMA and QuickTime protocols
  - delivering full duplex audio and videophone functions for broadband networks
  - uses a scalable bit rate and Ethernet port to enable operation over cable, xDSL, Ethernet, 3G and fiber
  - video capabilities of up to 30 frames/second performance in CIF resolutions and still images at 4CIF resolution

Blackfin Platforms

- **Blackfin BRAVO Platform**
  - enables any cable-connected standard NTSC/PAL TV (with RCA output connectors and camera input connectors) to a high-quality videophone
  - enables a PC browser to display what the camera sees, supporting an HTTP server function with MJPEG and MPEG4 video to enable access and control from any standard PC with a Microsoft Internet Explorer or Netscape browser or QuickTime player
  - optional trigger function enables the browser to sound an alarm when the camera senses predefined changes in motion or position

Targeted Application

- **Digital Media Processing**
  - Digital Camera
  - Camcorder
  - Video Capture Board
  - Document Scan/Fax
- **Portable Information Appliances**
  - video-enabled handheld devices (PDAs)
  - web phones/terminals
  - e-mail terminals

Targeted Application

- **Portable Information Appliances**
  - Internet audio players
- **Automotive Telematics, Infotainment, and Driver Assistance**
  - Telematics
  - Navigation/GPS
  - Car Audio Amplifiers
  - Hands Free
  - Voice Activated Control

Targeted Application

- **Networking and Internet Appliances**
  - VoIP Gateways
  - Multi-Service Applications
  - IP-PBX, PBX Adapters
  - Networked Set-Top-Box
  - Internet-smart handheld devices (PDAs)
  - Internet gaming devices
  - VoIP phones/terminals
  - NetTV

Targeted Application

- **Wireless**
  - Wireless Terminals
  - GSM Voice Wireless Handsets
  - GPRS Wireless Terminals
  - EDGE Wireless Terminals
  - TD-SCDMA Wireless Terminals
  - W-CDMA/UMTS Wireless Terminals

Targeted Application

- **Wireless**
  - W-CDMA/UMTS Wireless Terminals
  - Entertainment Wireless Terminals
  - Smart phone Wireless Terminals
  - 2.5G and 3G Wireless Base Stations

References

- **Blackfin Processor**
  - http://www.analog.com/processors/processors/blackfin/index.html
- **Getting Started With Blackfin Processor**
  - http://www.analog.com/UploadedFiles/Associated_Docs/356225839blk_ug_40.pdf
- **Blackfin Processor Architecture Overview**
  - http://www.analog.com/processors/processors/blackfin/blackfinArchOverview.html

References

- **Blackfin Processor Core Basics**
  - http://www.analog.com/processors/processors/blackfin/blackfinCoreBasics.html
- **Analog Devices ADSP-BF5xx**
  - http://www.analog.com/processors/processors/blackfin/pdf/blackfin_summary.pdf
- **Blackfin Target Market Backgrounder**
  - http://www.analog.com/processors/processors/blackfin/blackfinMarketsApplications.html

References

- **Blackfin Processor Instruction Set Reference**
  - http://www.analog.com/processors/epManualsDisplay/0,2795,,00.html?SectionWeblawId=207&ContentID=39274&Language=English
- **Blackfin Embedded Processor Data Sheet**
  - http://www.analog.com/UploadedFiles/Data_Sheets/144127970ADSP_BF531_2_3_a.pdf