ANALOG DEVICES

16 × 16-Bit CMOS Single Port Multiplier/Accumulator

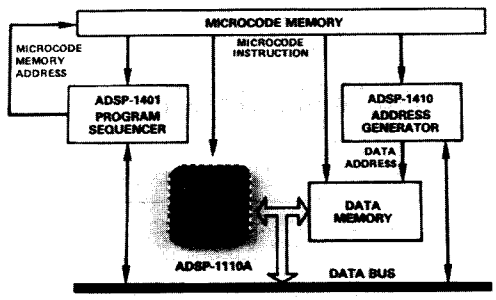
ADSP-1110A

FEATURES

16 × 16-Bit Parallel Multiplication/Accumulation
40-Bit Wide Accumulator with Overflow Flag, Saturation Arithmetic, and Shift-Left Control
Twos Complement or Unsigned Magnitude Inputs
85ns Multiply/Accumulate Time
28-Lead Ceramic DIP, Plastic DIP Package or Plastic Leaded Chip Carrier
350mW Power Dissipation with CMOS Technology
Specified Over the Extended Temperature Range
Pin-Compatible with ADSP-1110

APPLICATIONS

Digital Filtering
Fast Fourier Transforms
Matrix Multiplication
Microprocessor Acceleration



WORD-SLICE® MICROCODED SYSTEM WITH ADSP-1110A

GENERAL INFORMATION

The ADSP-1110A is a high-speed, low-power single-port 16 × 16-bit multiplier/accumulator (MAC), with processing throughput comparable to existing three-port MACs. Its single-bus structure offers unique advantages: more compact packaging in a 28-pin package, simpler system interface to single-bus peripherals, and significantly reduced cost. In addition, innovative on-chip features extend the ADSP-1110A's capabilities and eliminate external hardware.

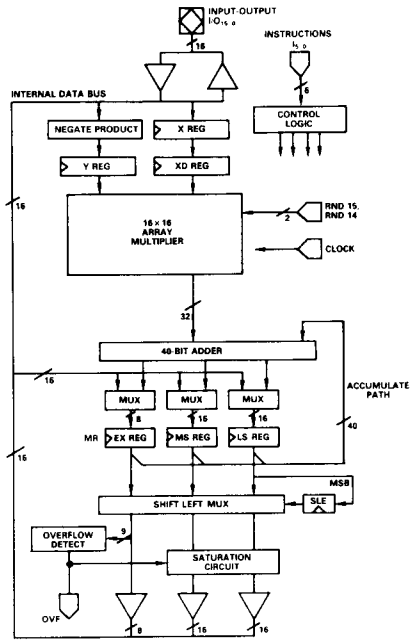
All inputs to and outputs from the ADSP-1110A pass through its single 16-bit I/O port. All I/O operations are single cycle. A multiplication or MAC operation requires two cycles to complete—consistent with the two cycles required to load input pairs to the multiplier. An internal pipeline register enables a new input to be loaded as the previous multiplication/accumulation is computed—allowing the device's full 11.7MHz computational bandwidth to be utilized.

A six-bit microcode instruction word governs the ADSP-1110A's operation. The instruction set centers around I/O and multiplication/accumulation operations. Additional instructions allow extra precision in single- and double-precision operations to be obtained efficiently.

Multiplier products are accumulated in a 40-bit wide Multiplier Result (MR) register, which consists of a 16-bit MS (Most Significant) and LS (Least Significant) register, and an 8-bit EX (Extension) register. Either multiplier input can be a twos complement or unsigned magnitude number. Overflow from the lower 32 bits of the MR into the upper eight guard bits is detected and can be monitored externally. Outputs can, conditional upon overflow status, be saturated to full scale. An MR register can be shifted left by one bit upon output; two independent controls allow rounding consistent with output formatting.

The ADSP-1110A is optimal for applications where board space is limited but the performance of a DSP processor is required. In addition, a microprocessor-based system can realize greater throughput by utilizing the ADSP-1110A in an accelerator.

5



ADSP-1110A Functional Block Diagram

SPECIFICATIONS[1]

RECOMMENDED OPERATING CONDITIONS

Parameter		ADSP-1110A J and K Grades Min	ADSP-1110A J and K Grades Max	ADSP-1110A S and T Grades[2] Min	ADSP-1110A S and T Grades[2] Max	Unit
V_{DD}	Supply Voltage	4.75	5.25	4.5	5.5	V
T_{AMB}	Operating Temperature ($T_{AMBIENT}$)	0	70	−55	125	°C

ELECTRICAL CHARACTERISTICS

Parameter		Test Conditions	ADSP-1110A J and K Grades Min	ADSP-1110A J and K Grades Max	ADSP-1110A S and T Grades[2] Min	ADSP-1110A S and T Grades[2] Max	Unit
V_{IH}	High-Level Input Voltage	@ V_{DD} = max	2.0		2.2		V
V_{IL}	Low-Level Input Voltage	@ V_{DD} = min		0.8		0.8	V
V_{OH}	High-Level Output Voltage	@ V_{DD} = min & I_{OH} = −1.0mA	2.4		2.4		V
V_{OL}	Low-Level Output Voltage	@ V_{DD} = min & I_{OL} = 4.0mA		0.4		0.6	V
I_{IH}	High-Level Input Current	@ V_{DD} = max & V_{IN} = 5.0V		10		10	μA
I_{IL}	Low-Level Input Current	@ V_{DD} = max & V_{IN} = 0V		10		10	μA
I_{OZH}	Three State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = max		50		50	μA
I_{OZL}	Three State Leakage Current	@ V_{DD} = max; High Z; V_{IN} = 0		50		50	μA
I_{DD}	Supply Current	@ max clock rate; TTL-inputs		70		80	mA
I_{DD}	Supply Current – Quiescent	All V_{IN} = 2.4V		35		40	mA

SWITCHING CHARACTERISTICS

Parameter		ADSP-1110A J Grade 0 to +70°C Min	J Grade Max	K Grade 0 to +70°C Min	K Grade Max	S Grade[2] −55°C to +125°C Min	S Grade[2] Max	T Grade[2] −55°C to +125°C Min	T Grade[2] Max	Unit
t_{CLK}	Clock Period	50		42.5		60		50		ns
t_{MAC}	Multiply/Accumulate Time		100		85		120		100	ns
t_{PW}	Clock Pulse Width	15		15		15		15		ns
t_{DS}	Input Data Setup Time	15		15		15		15		ns
t_{CS}	Input Control Setup Time	25		20		25		20		ns
t_{DH}	Input Data Hold Time	3		3		4		4		ns
t_{CH}	Input Control Hold Time	5		5		6		6		ns
t_D	Control to Valid Output		30		25		30		30	ns
t_{DSAT}	Control to Valid Output with Saturation		35		32		40		35	ns
t_{DIS}	Output Driver Disable Time		25		25		25		25	ns
t_O	Control to Overflow Flag		30		25		35		30	ns
t_{LO}	Control to Overflow Flag w/sl		40		35		45		40	ns

NOTES

[1]All min & max specifications are over power supply and temperature range indicated. Rise times are 5ns. Input levels are GND and 3.0V. Input timing reference levels and output reference levels are 1.5V.

[2]S and T grade parts are available processed and tested in accordance with MIL-STD-883, Class B. The processing and test methods used for S/883B and T/883B versions of the ADSP-1110A can be found in Analog Devices' Military Databook.

Specifications subject to change without notice.

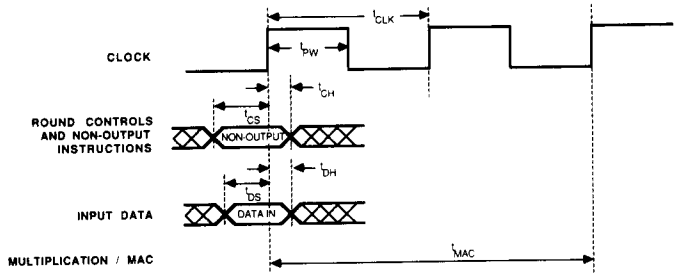


Figure 1a. ADSP-1110A Timing: Clocked (Synchronous) Operations All Non-Output Instructions

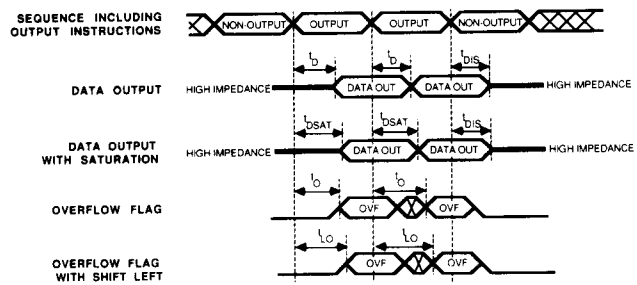


Figure 1b. ADSP-1110A Timing: Unclocked (Asynchronous) Operations All Output Instructions

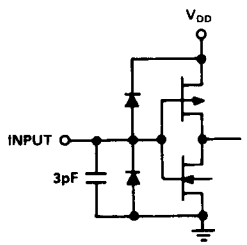


Figure 2a. Equivalent Input Circuits

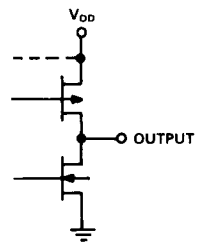


Figure 2b. Equivalent Output Circuits

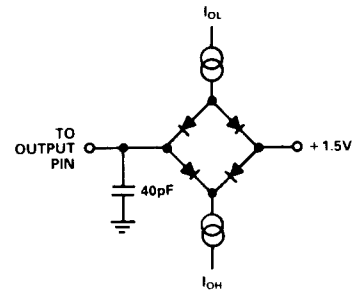


Figure 3. Normal Load Circuit for ac Measurements

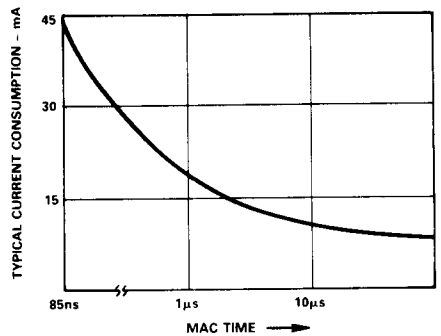


Figure 4. Typical Power Dissipation vs. Frequency

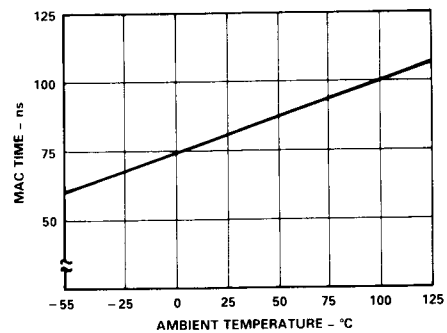


Figure 5. Typical Multiply Time vs. Temperature

METHOD OF OPERATION

The ADSP-1110A's operation is controlled by a six-bit microcode instruction and two rounding control pins. Table III presents instructions that are executed by the ADSP-1110A, along with the corresponding six-bit microcode instruction. The sections below further describe the instruction groups presented in Table III.

Input and Multi-Operation Instructions

A dedicated *input instruction* ("X = BUS") loads the X input at the rising edge of the clock. The X input is loaded with the data that is set up on the device's 16-bit I/O port.

A set of *multi-operation instructions* ("Y = BUS; CKMR; X*Y") are used to load the Y input and otherwise control the ADSP-1110A's multiplier/accumulator. Specifically, at the next rising clock edge, a multi-operation instruction i) loads Y input ii); clocks the result of the previous multiplication/MAC operation into the MR; and, iii) initiates the next multiplication/MAC operation. The multiplication/MAC operation is initiated at the rising edge of the clock and requires two cycles to complete. The instruction controls needed to govern the device's multiplier array and 40-bit adder during these two cycles are registered internally.

During the first cycle of a multi-operation instruction, the X input is transferred to an internal pipeline register (XD), and is latched there on the next rising clock edge. Consequently, a new X value can be loaded onto the chip during the second cycle of the multi-operation instruction. XD will not be overwritten until a new X value is loaded.

The ADSP-1110A supports the following multiplication and multiplication/accumulation operations:

$$\pm X*Y$$

and,

$$\pm X*Y \pm MR$$

The ADSP-1110A allows either input to be specified as a twos complement or unsigned magnitude number. Table II describes, for all combinations of inputs, the proper interpretation of the MR register if it is output with or without the left-shift option. Note that if the Y input is negative full scale and a negative product is specified, an invalid result is obtained. This happens because the ADSP-1110A will attempt to produce the unrepresentable twos complement of full-scale negative.

The result of a multiplication or MAC operation is latched into the MR register in either of two ways. A dedicated "CKMR" instruction performs this clocking. In addition, all multi-operation instructions clock the MR, eliminating overhead when computing MAC's (see *Instruction Sequences*). It is important to note that whenever "CKMR" is executed, it clocks the result of the *previous* operation into the MR. Also, in all cases, the clocking of the MR occurs at the rising edge of the clock.

MR Register Instructions

A number of the ADSP-1110A's instructions affect the contents of the MR register—including *preload instructions*, *transfer instructions*, and *sign extend instructions*. In addition, special output instructions allow for format adjusting the MR upon output.

The 40-bit accumulator of the ADSP-1110A is segmented into three registers: a 16-bit most significant product register (MS); a 16-bit least significant product register (LS); and, an 8-bit extended product register (EX) (see Table II). The eight guard bits of the EX allow at least 256 multiplication/accumulations without risk of overflow.

Dedicated instructions allow any of the MR's registers to be preloaded with data set up on the device's 16-bit I/O port. This preloading occurs at the rising edge of the clock.

The proper sequence for preloading a value Z into MR and adding it to the product $X_1 * Y_1$ is:

Instruction	Comment
1. X = BUS	Load X_1
2. Y = BUS; CKMR; X*Y + MR	Load Y_1; clock garbage into MR; initiate MAC
3. LS = BUS	Preload MR with Z
4. MS = BUS	Preload MR with Z
5. EX = BUS	Preload MR with Z
6. X = BUS	Load X_2
7. Y = BUS; CKMR; X*Y + MR	Load Y_2; MR = $X_1*Y_1 + Z$; initiate next multiplication.

This sequence ensures that the value Z preloaded by instructions 3, 4, and 5 is added to the product X_1*Y_1 and clocked into MR by instruction 7. If Z were preloaded prior to instruction 2, then instruction 2's "CKMR" operation would overwrite the Z value with the product of whatever values were last placed in the multiplier array.

Transfer operations allow one MR register to be moved down to an adjacent one—useful in double-precision operations. The ADSP-1110A can, in one cycle, shift the EX to the MS or the MS to the LS register. The shift left extend register (SLE) is a one-bit latch that is loaded with the value of the MSB of the LS register whenever the MS is transferred to LS. The SLE register retains its value until the next downshift of MS into LS overwrites its contents.

Anytime the result of a multiplication or multiplication/accumulation operation is clocked into the accumulator, the result is automatically sign extended into the upper MSBs of the accumulator. In addition, explicit instructions allow the MSB of the LS to be sign extended to the MS ("MS = SIGN EXT LS") or the MSB of the MS to be sign extended to the EX register ("EX = SIGN EXT MS"). Such sign extend capability may be needed to properly initialize the MR after the MS or LS is preloaded, or after an MR register transfer.

Output Instructions

Output instructions allow any MR register to be read. When written onto the ADSP-1110A's 16-bit bus, the 8-bit EX register is automatically sign-extended into the upper 8 MSBs of the bus. Standard output instructions of the ADSP-1110A are supplemented with two important options: a shift-left capability and conditional saturation.

The ADSP-1110A's output instructions include the ability to shift any MR register (EX, MS, or LS) left by one bit upon output. This shift does not affect the contents of MR, but does affect what appears on the ADSP-1110A's 16-bit I/O port. Figure 6 shows which bits of the 40-bit wide MR register are output if the shift-left option is invoked.

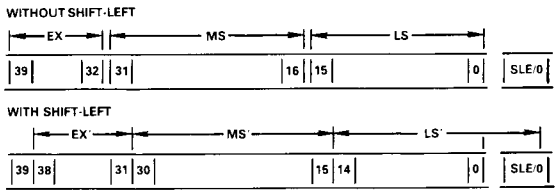


Figure 6. Effect of Left Shift on MR Outputs

The shift left-on-output control, which scales up the MR outputs by a factor of two, is useful under many circumstances. Twos complement multiplication—for all but one case (negative full-scale times negative full-scale)—results in redundancy in the two MSBs of the 32-bit product. This redundancy means that the 16-bit MS register contains two identical sign bits (bits 31 and 30 of MR) and just 14 bits of magnitude. The ADSP-1110A's shift-left control allows full precision in twos complement operations to be attained. Left shift control also provides a means for maximizing resolution when using block floating point, when downscaling twos complement results, and when upscaling mixed and unsigned magnitude results.

Whenever the RND14 pin is asserted during a "BUS = LS (sl)" or "BUS = LS (sl, sat)" instruction, the SLE bit will be appended to the upper 15 bits of the shifted LS. If the RND14 is low, however, a zero will be inserted into the LSB of LS. Appending the SLE bit to the shifted LS provides an extra bit of precision in applications such as double-precision multiplication/accumulations.

Round Controls

The RND14 and RND15 pins are two independent controls that allow rounding consistent with shifted or unshifted outputs, respectively. The round control signals are latched at the rising clock edge whenever the device receives a multiplication or MAC instruction. Asserting the RND 15 (RND14) pin will cause a 1 to be added to bit 15 (bit 14) of the LS. The rounding will not occur until the subsequent cycle in which the result of the multiplication or MAC operation is clocked into the MR.

Overflow and Saturation

The ADSP-1110A's overflow flag monitors 9 bits (8 in the EX register and the MSB of the MS register). If any bits in EX differ from the MSB of MS, then an overflow has occurred from the MS into the EX, and the overflow flag is asserted (HI) following an output instruction. Generally, the status of the overflow flag reflects the current contents of the MR and is updated each time a new result is clocked into the MR. However, if the MS register is output with a left-shift, the overflow logic determines whether the shifted MS overflows into the EX (when bits 38 through 30 are not identical) and is set accordingly. On the cycle following any left-shifted output, the overflow flag status reverts to reflect the contents of the MR. During cycles when a non-output instruction is executed, the overflow flag is always LO.

Serious data glitches can result from wraparound effects due to overflow in long multiply/accumulate chains. For example, if a positive number is added to positive full-scale, the 32 MSBs of the MR register will overflow into the 8-bit EX register. Simply reading the MS register will yield a negative twos complement number. To prevent this wraparound, the ADSP-1110A can—conditioned on overflow status—saturate an output to twos-complement full scale.

The ADSP-1110A's saturation logic operates only on output values; it has no effect on the contents of the MR register. This logic examines the sign of the MR (bit 39, the MSB of the EX register) and the overflow status. As Table IV indicates, the low 32 bits of the MR are saturated to full-scale positive (negative) if overflow has occurred in a positive (negative) MR.

Either the MS or LS registers can be left-shifted on output with conditional saturation. If the shifted value overflows the lower 32 bits, the outputted result will be saturated to full scale.

While the saturation control protects against overflow from the MS to the EX register, the user is not protected in the event the accumulated result overflows the entire 40-bit MR register.

5

OVF	MR BIT 39 (SIGN BIT)	Output Value with Saturation MS	LS
0	0	← No Change →	
0	1	← No Change →	
1	0	0111111111111111	1111111111111111
1	1	1000000000000000	0000000000000000

Table I. Overflow and Saturation Circuitry Conditions

	MR			
INPUTS (X & Y)	EX (8 BITS)	MS (16 BITS)	LS (16 BITS)	
b_{15} b_{14} ..... b_0	b_{39} b_{38} ..... b_{32}	b_{31} b_{30} ..... b_{16}	b_{15} ..... b_1	b_0
TWOS COMPLEMENT INTEGER -2^{15} 2^{14} ..... 2^0	(w/o sl) -2^{39} 2^{38} ..... 2^{32}	2^{31} ..... 2^{16}	2^{15} ..... 2^1	2^0
	(w/ sl) -2^{38} 2^{37} ..... 2^{31}	2^{30} ..... 2^{15}	2^{14} ..... 2^0	SLE/0
TWOS COMPLEMENT FRACTIONAL -2^0 2^{-1} ..... 2^{-15}	(w/o sl) -2^8 2^8 ..... 2^2	2^1 2^0 ..... 2^{-14}	2^{-15} ..... 2^{-29}	2^{-30}
	(w/ sl) -2^8 2^7 ..... 2^1	2^0 2^{-1} ..... 2^{-15}	2^{-16} ..... 2^{-30}	SLE/0
UNSIGNED MAGNITUDE INTEGER 2^{15} ..... 2^0	2^{39} ..... 2^{32}	2^{31} ..... 2^{16}	2^{15} ..... 2^1	2^0
UNSIGNED MAGNITUDE FRACTIONAL 2^{-1} ..... 2^{-16}	2^7 ..... 2^0	2^{-1} ..... 2^{-16}	2^{-17} ..... 2^{-31}	2^{-32}
MIXED MODE INTEGER -2^{15} 2^{14} ..... 2^0 & 2^{15} 2^{14} ..... 2^0	-2^{39} 2^{38} ..... 2^{32}	2^{31} ..... 2^{16}	2^{15} ..... 2^1	2^0
MIXED MODE FRACTIONAL -2^0 2^{-1} ..... 2^{-15} & 2^{-1} 2^{-2} ..... 2^{-16}	-2^8 2^7 ..... 2^1	2^0 ..... 2^{-15}	2^{-16} ..... 2^{-30}	2^{-31}

Table II. ADSP-1110A Data Formats

Instruction Group	Instruction	Microcode Instruction 5 4 3 2 1 0	Comments
Miscellaneous	NOP	0 0 0 0 x x	No Operation
	CKMR	0 0 0 1 x x	Clock MR
Input	X = BUS	0 0 1 0 x x	
Preload	LS = BUS	0 1 0 0 0 0	
	MS = BUS	0 1 0 1 x 0	
	EX = BUS	0 1 0 0 1 0	
Transfer	LS = MS	0 1 0 0 0 1	Sets SLE register
	MS = EX	0 1 0 1 0 1	
Sign Extend	EX = SIGN EXT MS	0 1 0 0 1 1	
	MS = SIGN EXT LS	0 1 0 1 1 1	
Output	BUS = EX	0 0 1 1 0 1	All output instructions are asynchronous
	BUS = EX (sl)	0 0 1 1 0 0	I5 – I2:
	BUS = MS	0 1 1 0 0 1	0011 = EX
	BUS = MS (sl)	0 1 1 0 0 0	0110 = MS
	BUS = MS (sat)	0 1 1 0 1 1	0111 = LS
	BUS = MS (sl,sat)	0 1 1 0 1 0	I1 – I0:
	BUS = LS	0 1 1 1 0 1	01 = to bus
	BUS = LS (sl)	0 1 1 1 0 0	00 = to bus shifted
	BUS = LS (sat)	0 1 1 1 1 1	10 = to bus shifted w/saturation
	BUS = LS (sl,sat)	0 1 1 1 1 0	11 = to bus w/saturation
Multi-Operation	Y = BUS; CKMR; $X_{US}*Y_{US}$	1 0 0 x 0 0	Require two cycles to complete.
	Y = BUS; CKMR; $-X_{US}*Y_{US}$	1 0 0 x 0 1	Other instructions can be executed
	Y = BUS; CKMR; $X_{US}*Y_{US}$ +MR	1 0 0 0 1 0	on the second cycle.
	Y = BUS; CKMR, $-X_{US}*Y_{US}$ +MR	1 0 0 0 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{US}$ −MR	1 0 0 1 1 0	I5 = Multiplication/MAC operation
	Y = BUS; CKMR; $-X_{US}*Y_{US}$ −MR	1 0 0 1 1 1	I4 = Y twos complement
	Y = BUS; CKMR; $X_{TC}*Y_{US}$	1 0 1 x 0 0	I3 = X twos complement
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$	1 0 1 x 0 1	I2 = Subtract previous result
	Y = BUS; CKMR; $X_{TC}*Y_{US}$ +MR	1 0 1 0 1 0	I1 = Add/subtract previous result
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$ +MR	1 0 1 0 1 1	from product
	Y = BUS; CKMR; $X_{TC}*Y_{US}$ −MR	1 0 1 1 1 0	I0 = Negate product
	Y = BUS; CKMR; $-X_{TC}*Y_{US}$ −MR	1 0 1 1 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$	1 1 0 x 0 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$	1 1 0 x 0 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$ +MR	1 1 0 0 1 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$ +MR	1 1 0 0 1 1	
	Y = BUS; CKMR; $X_{US}*Y_{TC}$ −MR	1 1 0 1 1 0	
	Y = BUS; CKMR; $-X_{US}*Y_{TC}$ −MR	1 1 0 1 1 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$	1 1 1 x 0 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$	1 1 1 x 0 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$ +MR	1 1 1 0 1 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$ +MR	1 1 1 0 1 1	
	Y = BUS; CKMR; $X_{TC}*Y_{TC}$ −MR	1 1 1 1 1 0	
	Y = BUS; CKMR; $-X_{TC}*Y_{TC}$ −MR	1 1 1 1 1 1	

Mnemonic Definitions

=	Assign right side to left.	sl	Shift left.
BUS	16-bit external data bus used for all I/O operations.	sat	Conditional on overflow, saturate the outputted value.
X	Input register for multiplier.	TC	Two's complement number.
Y	Input register for multiplier.	US	Unsigned magnitude number.
EX	8-bit extension register for accumulator.	SIGN	Sign bit (MSB) of specified register.
MS	16-bit most significant product register.	CKMR	Clock product into EX, MS, and LS.
LS	16-bit least significant product register.	*	Multiply
MR	40-bit accumulator comprising EX, MS and LS.	x	Microcode instruction bit can be either a 0 or 1.

Table III. ADSP-1110A Instruction Set

CLOCK AND TIMING

Figure 1 presents a timing diagram for the ADSP-1110A's operation.

Input data, round controls, and non-output instructions are clocked (synchronous); set-up and hold times are specified accordingly. All multi-operation (two cycle) instructions are clocked, and the internal controls needed for the second cycle are latched internally.

Unlike all other ADSP-1110A instructions, output operations are asynchronous. The relevant timing specification is the delay between control inputs and valid outputs. The use of saturation (sat) slows down the availability of a valid output on the ADSP-1110A's I/O bus; delay times are specified accordingly.

The ADSP-1110A's OVF (overflow) flag is set according to the contents of the MR register. However, upon outputting the MR with the shift-left control, the OVF flag may be modified if the left shift causes overflow. The relevant timing for this case is specified.

The ADSP-1110A's output three-state drivers are not disabled until t_{DIS} ns after an output instruction is removed. Since the ADSP-1110A has just one I/O port, bus contention can occur when an ADSP-1110A input immediately follows an output. For example, an input source (e.g., a data RAM) enabled to drive the bus immediately after an ADSP-1110A output creates the possibility that both drivers are active simultaneously. There are two ways to avoid such conflicts:

1. Set up output instructions well in advance of the clock's rising edge (>t_D set-up time), enabling the data output to complete in time for the data to be latched at the clock edge. Allow t_{DIS} ns after the clock edge before enabling a different device to drive the bus. Note that any system that provides the ADSP-1110A with its instruction from a pipeline register operates in this way. The *Hardware Implementations with the ADSP-1110A* section describes several alternative implementations consistent with this approach.
2. For systems with minimal instruction set-up time, an operation that doesn't use the bus (e.g., a NOP) may need to be inserted after an output instruction. The reason for this is as follows. Output instructions must be held valid for t_D ns, which means that—if instruction set-up time is minimal—output instructions must be held beyond the rising edge of the clock. After the output instruction is removed, another t_{DIS} elapses before the output drivers are disabled. As a result, the three-state output drivers are active well into the next cycle. If the bus is driven with an input in the next cycle, bus contention may occur.

Instruction Sequences

With the ADSP-1110A, single multiplication operations involve three overhead statements in addition to the multiply command, as Figure 7 illustrates.

While a multiplication/accumulation sequence is structurally similar to a single multiplication, overhead as a percentage of computation time is reduced substantially. In the instruction flow diagram shown in Figure 8, a NOP is needed only in the final multiplication/accumulation operation. Also, new X values are loaded as multiplication/accumulation instructions complete. In this sequence, the three cycles of overhead can be spread out over as many multiplication/accumulations as are performed consecutively.

For a series of multiplication/accumulation sequences, I/O operations can be further overlapped. At the end of each multiplication/accumulation string, a new string is initiated. In this instance, overhead cycles become negligible in importance; the multiplication/accumulation rate of the ADSP-1110A approaches 11MHz.

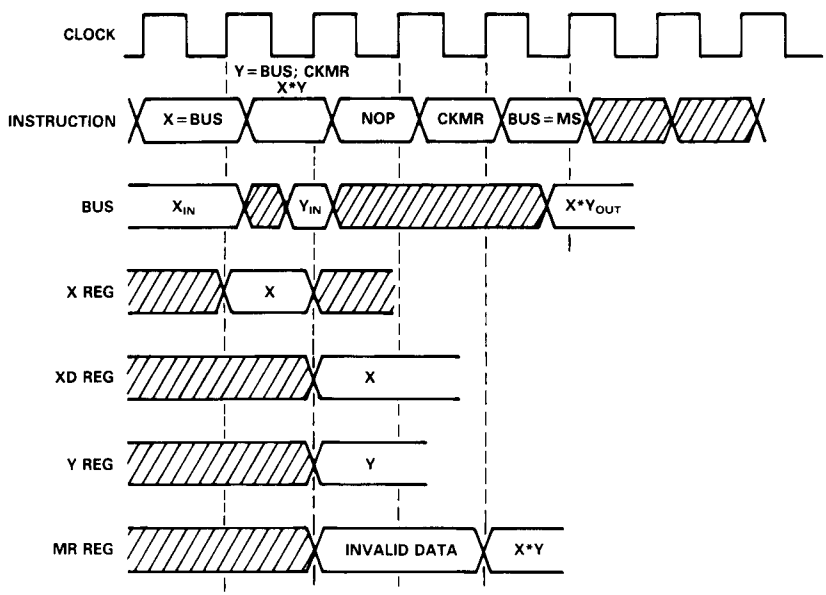


Figure 7. Multiply Operation Timing

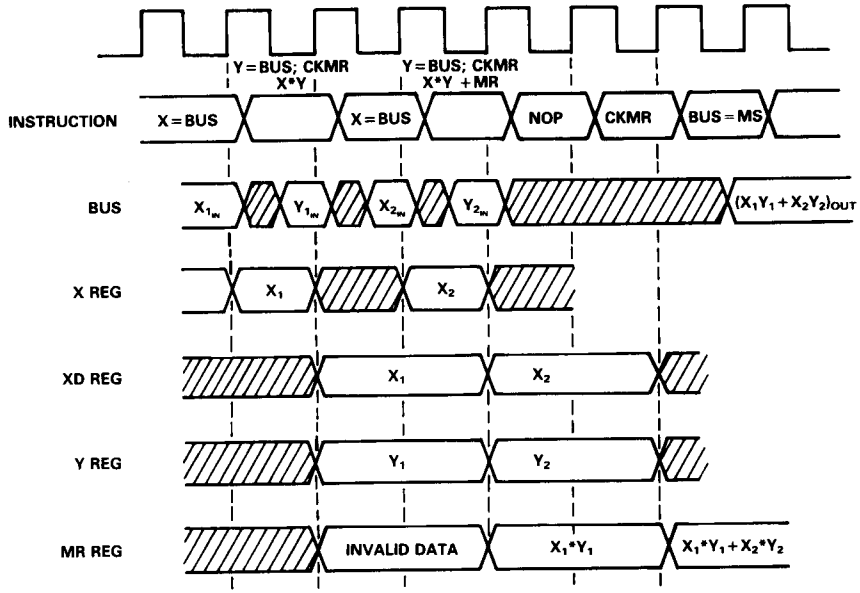


Figure 8. Multiply/Accumulate Operation Timing

Avoid Bus Contentions

Because the ADSP-1110A typically shares its data port with other devices on a common bus, there is a potential for bus contentions at power-up. If the instruction applied to the ADSP-1110A at power-up is random, the multiplier/accumulator could be in an output state. If any other devices are driving the bus at the same time, there will be a bus contention.

The obvious solution is to make sure no other devices are driving the common data bus at power-up. Another approach is to force instruction bit I_5 (pin 18) HI at power-up. This guarantees that the ADSP-1110A will not be in an output state because ADSP-1110A output instructions are asynchronous and all have a zero in instruction bit 5 (I_5).

HARDWARE IMPLEMENTATIONS WITH THE ADSP-1110A

There are many alternative ways of implementing high performance DSP systems with the ADSP-1110A. The following sections illustrate some of the more commonly used approaches using the ADSP-1110A: a microcoded system, a ROM-based sequential machine, a PLA-based state machine, and as a device directly interfaced to a microprocessor. The optimal implementation will depend on the performance, price, and board area requirements of the design.

Microcoded System

Many microcoded systems have the design objective of fast number crunching, while minimizing microcode bits and circuit board area. The ADSP-1110A single port MAC—with just 8 control bits, its single bus structure, and fast cycle time—helps meet these objectives. The ADSP-1110A can be simply connected to the processor data bus and microcode instruction field to provide powerful multiplier/accumulator functions.

A typical Word-Slice® processor with the ADSP-1110A is shown below:

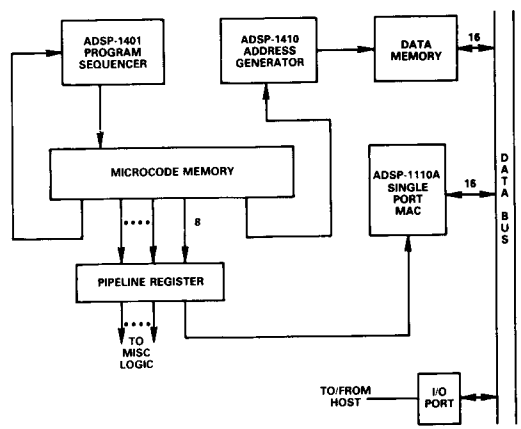


Figure 9.

In most bit-slice designs, the control bits from the microcode memory are latched in a pipeline register. In the above implementation, the ADSP-1110A and all miscellaneous logic are used in conjunction with an external pipeline latch. The pipeline latch guarantees that the microcode bits controlling the circuitry are valid for a complete cycle (see timing diagram below). Note that the ADSP Word-Slice® components (the ADSP-1401 and ADSP-1410) contain an internal pipeline register and are fed directly from the microcode.

Word-Slice is a registered trademark of Analog Devices, Inc.

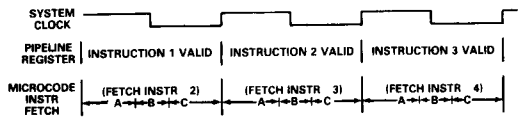


Figure 10.

Segments A, B and C of the above microcode instruction fetch cycle denote different operations. The sequencer starts execution of an instruction at the beginning of a cycle and, during segment A, calculates the microcode memory address for the next instruction. The output of the sequencer (microcode memory address) is valid at the start of segement B and the data is accessed during this segment. In segment C, the data from the microcode memory is valid and is latched into the pipeline register at the end of the cycle (the rising edge of the clock). This rising clock edge is used to latch all registers and devices in the circuit.

Notice from the timing diagram that the instructions in the pipeline register are valid during the complete cycle. Also, while an instruction in the pipeline register is available to circuitry, the next instruction is concurrently being fetched from microcode memory to be subsequently latched into the pipeline register at the start of the next cycle.

To better understand how to program the ADSP-1110A in a pipelined architecture, the ADSP-1110A's instruction set can be divided into three functional classes. The first includes all instructions that cause a register to be loaded. The second class includes output instructions, which are asynchronous. The final class of instructions are the multiply operations. Note that multiply instructions perform multiple operations—they also load the Y register and clock MR.

Class 1 Register Loads	Class 2 Output	Class 3 Multiply
CKMR	Bus = EX	X*Y
X = Bus	Bus = EX (sl)	(All forms)
LS = Bus	Bus = MS	
MS = Bus	Bus = MS (sl)	
EX = Bus	Bus = MS (sat)	
LS = MS	Bus = MS (sl,sat)	
MS = EX	Bus = LS	
EX = SIGN EXT MS	Bus = LS (sl)	
MS = SIGN EXT LS	Bus = LS (sat)	
	Bus = LS (sl,sat)	

Table IV.

Register loads occur at the end of the cycle (rising clock edge) whenever a Class 1 instruction is presented. Output instructions (Class 2) are executed during the same cycle as presented, with the output data becoming valid t_D ns into the cycle and remaining valid throughout the rest of the cycle. This data is available to be latched into an external register, or other device, at the end of the cycle (rising clock edge).

Multiply instructions (Class 3) begin executing at the beginning of the cycle *after* the cycle in which the instruction is presented. These instructions require two cycles to complete. Therefore, when programming the ADSP-1110A with a multiply instruction, it must be noted that the instruction will not start execution until the next cycle, as opposed to Class 1 and 2 instructions, which are executed in the current cycle. This can be illustrated by the following program example.

Cycle	Pipeline Instruction	ADSP-1110A Activity
1	X = BUS	X register loaded with data at end of cycle.
2	Y = BUS; CKMR; X*Y	Y register loaded with data at end of cycle and multiplier control signals latched at end of cycle. Garbage clocked into the MR.
3	X = BUS	Multiply of first operands begins at start cycle, X register loaded with new data at end of cycle.
4	Y = BUS; CKMR; X*Y + MR	Y register loaded with new data. MR loaded with first product and multiplier control signals are latched at end of cycle.
5	NOP	Multiply of second operands begins at start of cycle.
6	CKMR	The sum of the second product and the old MR contents are loaded into the MR at the end of the cycle.
7	BUS = MS	Most significant portion of MR is output to the bus and data is valid t_D ns max into cycle.

Table V.

ROM-Based Sequential Machine

In a similar manner to which the microcode memory of the bit-slice machine provides control bit to circuit components, a ROM can be used in conjunction with a binary counter and a latch to provide these same control bits. Such an approach offers a more compact design, at the expense of versatility. A binary counter is used to sequence through ROM locations, thus implementing a specified algorithm. This architecture is shown below.

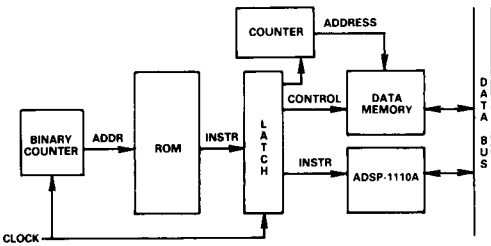


Figure 11.

The ROM contains the necessary bit patterns to control the miscellaneous circuitry and the ADSP-1110A. The binary counter provides sequential addresses to the ROM, resulting in the execution of a specific algorithm. The output of the ROM is latched so that the bits remain stable during the ROM access time associated with the next cycle. A separate counter is used to address the data memory.

This design technique can be expanded to access several functions stored in ROM by using a preloadable counter. The counter is preloaded with the starting address of the desired program in ROM. A dedicated control bit from the ROM is used to flag the counter, denoting the end of the program segment. Note that a latch is placed in front of the preloadable counter so that a host microprocessor can feed the required starting addresses if desired. This design is illustrated below.

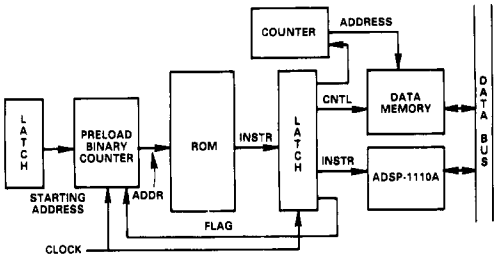


Figure 12.

PLA-Based State Machine

Instead of using the purely sequential approach of the ROM/counter solution, a latched PLA can generate the microcode. This state machine reduces real estate by entirely eliminating the latch (which is internal to the PLA) and the counter. No counter is needed since, in a state machine, the next output state is determined by the current output state. Use of state diagrams and CAD techniques provide the PLA truth table. However, in designs that require many states and complex state diagrams, a ROM-based sequential machine may be easier to implement.

Interfacing the ADSP-1110A to a Microprocessor

Because of its high speed, the ADSP-1110A can be used in an accelerator for microprocessors such as the Intel 286 or the Motorola 68000, performing dedicated macro-routines. The host microprocessor communicates with the ADSP-1110A via memory-mapping or I/O mapping (depending on the microprocessor). Also, the microprocessor and the ADSP-1110A must both have access to data memory. The microprocessor merely triggers the ADSP-1110A circuit and proceeds to perform some other task while the ADSP-1110A executes its macro-routine such as a matrix multiply, inverse function, square-root function, or digital filter. The following diagram illustrates a typical interface to a microprocessor.

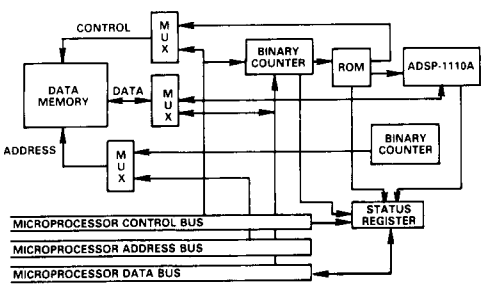


Figure 13.

Any signal that must communicate with the microprocessor is connected to the external status register. The status register is either I/O mapped or memory mapped. The overflow line from the ADSP-1110A, along with the end of program control flag, is also connected to the status register. Also, flags may be used to interrupt the microprocessor. Note that the high-speed data memory is available to both the microprocessor and the ADSP-1110A circuit. The multiplexers performing this selection are controlled by the microprocessor, with multiplexer select lines coming from the external status register.

APPLICATIONS

The ADSP-1110A is a high-performance component for a host of digital signal processing applications including FFT's, digital filters, and double-precision multiplication.

FFT Applications

The fast Fourier transform (FFT) is the principal algorithm used to analyze the frequency content of a signal. The FFT significantly reduces the time required to compute a Fourier transform by taking advantage of patterns in the computations to economize on multiplications. The ADSP-1110A performs the "butterfly," the key arithmetic operation in an FFT, entirely on-chip.

Figure 14 illustrates a decimation-in-time butterfly. As outlined by equations (1)

$$A_0' = A_0 + A_1 e^{j\theta} \qquad (1)$$
$$A_1' = A_0 - A_1 e^{j\theta}$$

the complex number A_1 is multiplied by a rotation term, $R = e^{j\theta}$, and added to the complex number A_0, producing A_0'. A_1' is obtained by subtracting the complex product A_1*R from A_0. The rotation R can be written:

$$R = e^{j\theta} = \cos\theta + j\sin\theta \equiv C + jS \qquad (2)$$

In an FFT A_0 and A_1 are complex numbers. Let

$$A_0 = X_0 + jY_0 \qquad (3)$$
$$A_1 = X_1 + jY_1$$

Then,

$$(A_1)e^{j\theta} = (X_1 + jY_1)(C + jS) \qquad (4)$$
$$= (X_1C - Y_1S) + j(X_1S + Y_1C)$$

allowing A_0' and A_1' to be represented as:

$$A_0' = X_0 + jY_0 + [(X_1C - Y_1S) + j(X_1S + Y_1C)] \qquad (5)$$
$$= X_0' + jY_0'$$
$$A_1' = X_0 + jY_0 - [(X_1C - Y_1S) + j(X_1S + Y_1C)]$$
$$= X_1' + jY_1'$$

Expanding and equating real and imaginary terms yields:

$$X_0' = X_0 + (X_1C - Y_1S) \qquad (6)$$
$$Y_0' = Y_0 + (X_1S + Y_1C)$$
$$X_1' = X_0 - (X_1C - Y_1S)$$
$$Y_1' = Y_0 - (X_1S + Y_1C)$$

Equations 6 can be used by the ADSP-1110A to efficiently implement the butterfly computation. First, X_0 is loaded into

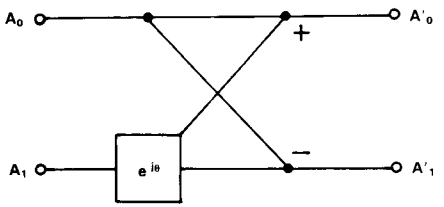


Figure 14. FFT "Butterfly" Diagram

MR by multiplying it by positive full scale (k≡0111...1). ("+1" cannot be represented in fractional two's complement, though "−1" can be. Scaling all terms by the factor "0.111111111111111" [binary] fits all values of sine and cosine into fractional two's complement and introduces less error than consistently failing to represent positive unity.) X_0' is obtained as follows:

$$X_0' = kX_0 + X_1C - Y_1S \quad (7)$$

Note that the factor k is not equal to unity. To ensure consistency in results, all stored cosine and sine factors (C and S) should similarly be scaled by k. Then, X_1' can be simply computed:

$$X_1' = kX_0 - (X_1C - Y_1S) \quad (8)$$
$$= 2kX_0 - X_0'$$

where X_0' is the accumulator's contents, and $2kX_0$ results from a mixed-mode multiplication of 2k and X_0. This operation represents a multiply/subtract, which illustrates an additional feature of the ADSP-1110A. Conventional MAC's cannot perform mixed-mode multiplies or multiply/subtracts.

Table VI provides the details for computing an FFT Butterfly with the ADSP-1110A. Each point requires ten cycles to compute the real component and ten cycles for the imaginary component. A 1024-point FFT requires 5120 butterflies.

A butterfly calculation contains a series of multiply/accumulates and multiply/subtracts. This presents a challenge in rounding the result, because rounded outputs from earlier cycles become inputs in later cycles. The rounding on cycles 4, 6, 14, and 16 ensures that the outputs (lines 7, 10, 17, and 20) are rounded correctly. Lines 4 and 14 round on bit 14, consistent with a left shift during output. However, lines 6 and 16 round on bit 15 to arrive at X_1' and Y_1'. In performing the multiply and subtract on these lines, the original round on lines 4 and 14 becomes inverted. To compensate, 2 must be added to the 14th bit, 1 to compensate for the previous round, and then 1 to round the current result. This can be easily accomplished in one step by adding a 1 to bit 15 rather than 2 to bit 14.

5

Cycle	Instruction	Comments
18'.	X = BUS	Load k
19'.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load X_0, MR = Previous
20'.	BUS = MS(sl)	Output Y_1' from previous butterfly
1.	X = BUS	Load C
2.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR	Load X_1, MR = kX_0
3.	X = BUS	Load S
4.	Y = BUS;CKMR; − X_{TC}*Y_{TC} + MR/RND14	Load Y_1, MR = $kX_0 + X_1C$
5.	X = BUS	Load 2k
6.	Y = BUS;CKMR;X_{US}*Y_{TC} − MR/RND15	Load X_0, MR = $kX_0 + (X_1C - Y_1S)$ + RND14
7.	BUS = MS(sl)	Output X_0'
8.	X = BUS	Load k
9.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load Y_0, MR = $kX_0 - (X_1C - Y_1S)$ + RND14
10.	BUS = MS(sl)	Output X_1'
11.	X = BUS	Load S
12.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR	Load X_1, MR = kY_0
13.	X = BUS	Load C
14.	Y = BUS;CKMR;X_{TC}*Y_{TC} + MR/RND14	Load Y_1, MR = $kY_0 + X_1S$
15.	X = BUS	Load 2k
16.	Y = BUS;CKMR;X_{US}*Y_{TC} − MR/RND15	Load Y_0, MR = $kY_0 + (X_1S - Y_1C)$ + RND14
17.	BUS = MS(sl)	Output Y_0'
18.	X = BUS	Load k
19.	Y = BUS;CKMR;X_{TC}*Y_{TC}	Load new X_0, MR = $kY_0 - (X_1S - Y_1C)$ + RND14
20.	BUS = MS(sl)	Output Y_1'

Table VI. Sample FFT "Butterfly" Sequence

FIR Filters

The ADSP-1110A is readily included in an FIR filter configuration. Figure 15 diagrams an N-tap finite impulse response (FIR) filter. FIR filters perform convolution in the time domain, corresponding to multiplication in the frequency domain. The coefficients h_i represent the filter's impulse response—the time domain equivalent of the filter's desired frequency response.

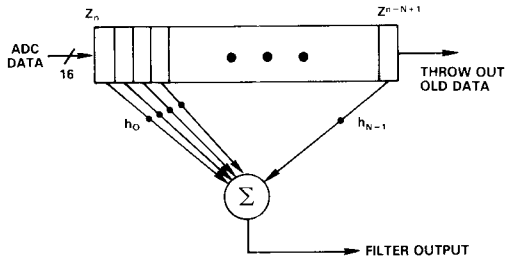


Figure 15. FIR Filter Design

When implemented with the ADSP-1110A, FIR filters employ a single RAM with a memory map as diagrammed in Figure 16. This contrasts with the multiple RAMs usually required in three-port MAC designs. Except in adaptive filters, where the filter response changes to meet changing system requirements, the filter coefficients remain constant.

Input data, on the other hand, is continuously updated. Each new data sample overwrites the oldest data point in RAM, an action that is tracked by an address counter. The Z_{n-N} sample, for instance, is overwritten with the new Z_n sample, and data points are addressed as a circular buffer.

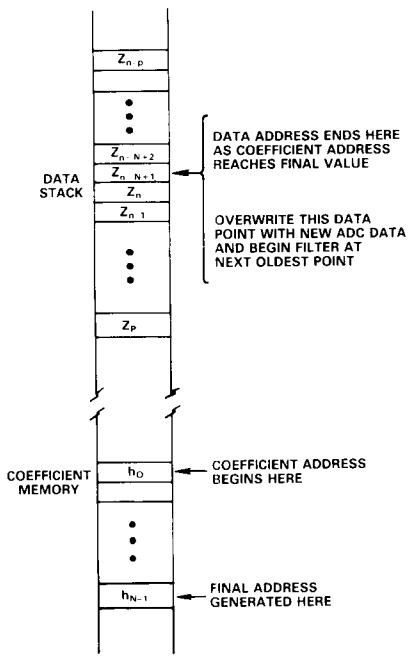


Figure 16. Memory Map

In the time interval between new data samples, the filter multiplies each of the N previously stored data samples, Z_i, by the respective filter coefficients, h_{n-i}. The resulting sum of the products represents the filtered signal output. An overflow flag and optional saturation logic allow long FIR filters to be implemented without risking overflow.

Note that the use of the ADSP-1110A does not require a complicated control circuit. Figure 17, for example, diagrams the controller circuit flow-chart for the FIR filter described above. When implemented in hardware, the controller's complexity remains comparable to that of the controller required for a three-port MAC-based filter design.

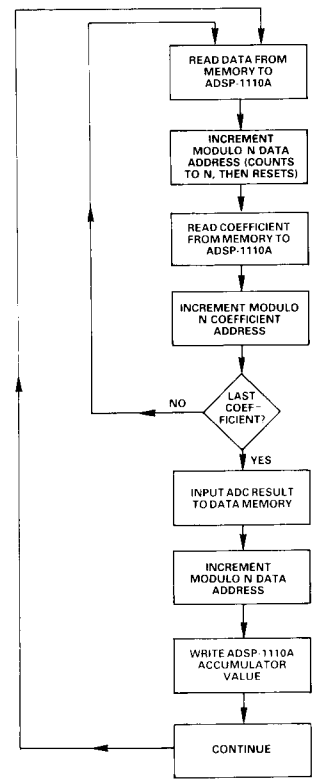


Figure 17. Controller Flow Chart for FIR Filter

IIR Filters

Infinite impulse reponse (IIR) filters use feedback to improve filter performance at the cost of a more complicated design. The principal advantage of an IIR filter is the relatively small number of multiplies needed to achieve a high-performance filter. The ADSP-1110A, unlike conventional MAC's, has architectual features that eliminate some of the disadvantages associated with implementing IIR's.

The time required for the ADSP-1110A to calculate a biquad section of an IIR filter is (5 MAC operations) × (two cycles/operation) + (output cycle), or eleven cycles. In the equation for a biquad,

$$Y_0 = a_0X_0 + a_1X_{-1} + a_2X_{-2} - b_1Y_{-1} - b_2Y_{-2} \qquad (9)$$

the coefficient b_1 generally lies between 1 to 2. The most conventional way to represent the coefficients and data is in fractional two's complement notation. Since this numbering system only ranges from -1 to 0.999 . . . , all coefficients and data for the IIR filter have to be divided by 2 to handle b_1 when using a conventional MAC. To compensate, external shifters are needed on output to shift the result up by one bit (multiply by 2).

A coefficient in the $+2$ to -2 range can be handled with the ADSP-1110A by using a mixed-mode multiply. Since the coefficient's sign is known in advance, multiply/add and multiply/subtract operations can supply the sign to an unsigned magnitude number. The MS register is left-shifted as usual on output to obtain the correct result.

Stability is an important issue for IIR filters. The ADSP-1110A's wide accumulator, together with the hard-limiting provided by its saturation circuit, prevent the overflow problems and large-scale oscillations that often plague IIR filters.

Double-Precision Multiplies

In order to handle double-precision multiplication (multiplying two 32-bit two's complement numbers), conventional MACs require additional external logic. The ADSP-1110A, in contrast, performs these operations without external support. Moreover, the device performs a double-precision multiplication in fifteen cycles, seven of which represent overhead.

Equation 10 represents a double-precision multiply:

$$\begin{aligned} P &= (X)(Y) \\ &= (MSW_x + LSW_x \cdot 2^{-16})(MSW_y + LSW_y \cdot 2^{-16}) \\ &= MSW_x \cdot MSW_y + (MSW_x \cdot LSW_y + MSW_y \cdot LSW_x) \cdot 2^{-16} \\ &\quad + LSW_x \cdot LSW_y \cdot 2^{-32} \end{aligned} \tag{10}$$

where P is the 64-bit product of two 32-bit two's complement numbers, X and Y. MSW_x represents the 16 most significant bits of word X, and LSW_x represents the 16 least significant bits. The product P equals the sum of partial products; each partial product's sign and significance must be taken into account in order to obtain the proper result.

A double-precision multiply requires no external logic. Furthermore, as illustrated in the following sequence, the ADSP-1110A performs the operation in 15 cycles.

In this double-precision multiply sequence, shown in Table VII, the four basic multiplications require only eight cycles. Cycles 5, 6, 11, 12, 13, 14, and 15 represent overhead.

Double-Precision MACs

The previous discussion concerned double-precision multiplies. The ADSP-1110A also readily handles double-precision multiply/accumulate operations. For example:

$$AP = \sum_{i=1}^{N} X_i Y_i \tag{11}$$

$$= \sum_{i=1}^{N} (MSW_{xi} + LSW_{xi} \cdot 2^{-16})(MSW_{yi} + LSW_{yi} \cdot 2^{-16})$$

Cycle	Operation	Comments
1.	X = BUS	Load LSW_x.
2.	Y = BUS;CKMR; X_{US}*Y_{US}/RND15	Load LSW_y and multiply (unsigned).
3.	NOP	No op.
4.	Y = BUS;CKMR; X_{US}*Y_{TC} + MR	Load MSW_y and perform MAC (mixed-mode).
5.	LS = MS	MS shifts into LS. The LS of the (LSW_x)(LSW_y) product is discarded.
6.	MS = EX	Shift EX into MS.
7.	X = BUS	Load MSW_x.
8.	Y = BUS;CKMR; X_{TC}*Y_{US} + MR/ RND14	Load LSW_y and perform MAC (mixed-mode) with round in bit 14.
9.	NOP	No op.
10.	Y = BUS;CKMR X_{TC}*Y_{TC} + MR	Load MSW_y and perform MAC (two's complement).
11.	LS = MS	Shift MS into LS.
12.	MS = EX	Shift EX into MS.
13.	CKMR	Clock the output registers. This loads the MAC from cycle 10 into the accumulator.
14.	BUS = MS(sl)	Output MS with left shift.
15.	BUS = LS(sl)/ RND14	Output LS with left shift. The SLE register provides an extra bit of precision.

Table VII.

5

where each X and Y is a 32-bit number, and AP is a 72-bit accumulated product. Note that AP can be expressed as the sum of accumulated partial products as follows:

$$\begin{aligned} AP = &\left[\left\{\sum_{i=1}^{N} (MSW_{xi})(MSW_{yi})\right\}\right] + \\ &+\left[\left\{\sum_{i=1}^{N} ((MSW_{xi})(MSW_{yi}) + (LSW_{xi})(MSW_{yi}))\right\}\right] \cdot 2^{-16} \\ &+\left\{\sum_{i=1}^{N} (LSW_{xi})(LSW_{yi})\right\} \cdot 2^{-32} \end{aligned} \tag{12}$$

Computing the accumulated double-precision product AP requires the same basic sequence as in computing a single-precision MAC. Simply compute a summation of partial products, rather than the summation of products themselves.

The summation of partial products often leads to sums greater than 32-bits. The 8-bit extension register stores any overflow, letting the summation proceed without error. The output and shift cycles occur once each at the end of the appropriate partial product calculation. A 32-point double-precision FIR filter, requires (32-points)(4-multiplies)(2 cycles/multiply) + 9 overhead cycles = 273 total cycles.

An optional procedure, which cuts the multiply/accumulate time by roughly 25%, entails omitting the $LSW_x \times LSW_y$ accumulation and instead adding ¼ of the number of accumulations to the final result. This removes the bias because the LSW's of both words have a mean value of ½ and when multiplied together have a mean product of ¼. Thus any bias in the answer is removed. A simple way to add 1's to the LSB is to assert the round control on the appropriate number of MAC operations.

ABSOLUTE MAXIMUM RATINGS

Supply Voltage	−0.3V to 7V
Input Voltage	−0.3V to V_{DD}
Output Voltage Swing	−0.3V to V_{DD}
Operating Temperature Range (Ambient)	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (10 Seconds)	300°C

PIN DESIGNATIONS
All Packages

PIN	FUNCTION	PIN	FUNCTION
1	RND15	15	CLK
2	RND14	16	I_3
3	I/O_{14}	17	I_4
4	I/O_{12}	18	I_5
5	I/O_{10}	19	OVF
6	I/O_8	20	I/O_1
7	I/O_6	21	I/O_3
8	I/O_4	22	I/O_5
9	I/O_2	23	I/O_7
10	I/O_0	24	I/O_9
11	I_0	25	I/O_{11}
12	I_1	26	I/O_{13}
13	I_2	27	I/O_{15}
14	GND	28	V_{DD}

ORDERING INFORMATION

Part Number	Temperature Range	Package	Package Outline
ADSP-1110AJD	0 to +70°C	28-Pin Ceramic DIP	D-28A
ADSP-1110AKD	0 to +70°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ASD	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ATD	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ASD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110ATD/883B	−55°C to +125°C	28-Pin Ceramic DIP	D-28A
ADSP-1110AJN	0 to +70°C	28-Pin Plastic DIP	N-28A
ADSP-1110AKN	0 to +70°C	28-Pin Plastic DIP	N-28A
ADSP-1110AJP	0 to +70°C	28-Lead Plastic Leaded Chip Carrier	P-28
ADSP-1110AKP	0 to +70°C	28-Lead Plastic Leaded Chip Carrier	P-28

Contact DSP Marketing in Norwood concerning the availability of other package types.

ESD SENSITIVITY

The ADSP-1110A features proprietary input protection circuitry to dissipate high energy discharges (Human Body Model). Per Method 3015 of MIL-STD-883, the ADSP-1110A has been classified as a Class 1 device.

Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should be discharged to the destination socket before devices are removed. For further information on ESD precautions, refer to Analog Devices' *ESD Prevention Manual*.
