

SECTION 4

DATA ARITHMETIC LOGIC UNIT

This section describes the operation of the data arithmetic logic unit (ALU) registers and hardware. The data representation, rounding, and saturation arithmetic used within the data ALU are also presented. This section concludes with a discussion of the programming model.

4.1 OVERVIEW AND DATA ALU ARCHITECTURE

The DSP56000/DSP56001 central processor is composed of three execution units that operate in parallel. They are the data ALU, address generation unit (AGU), and the program control unit (see Figure 4-1). These three units are register oriented rather than bus

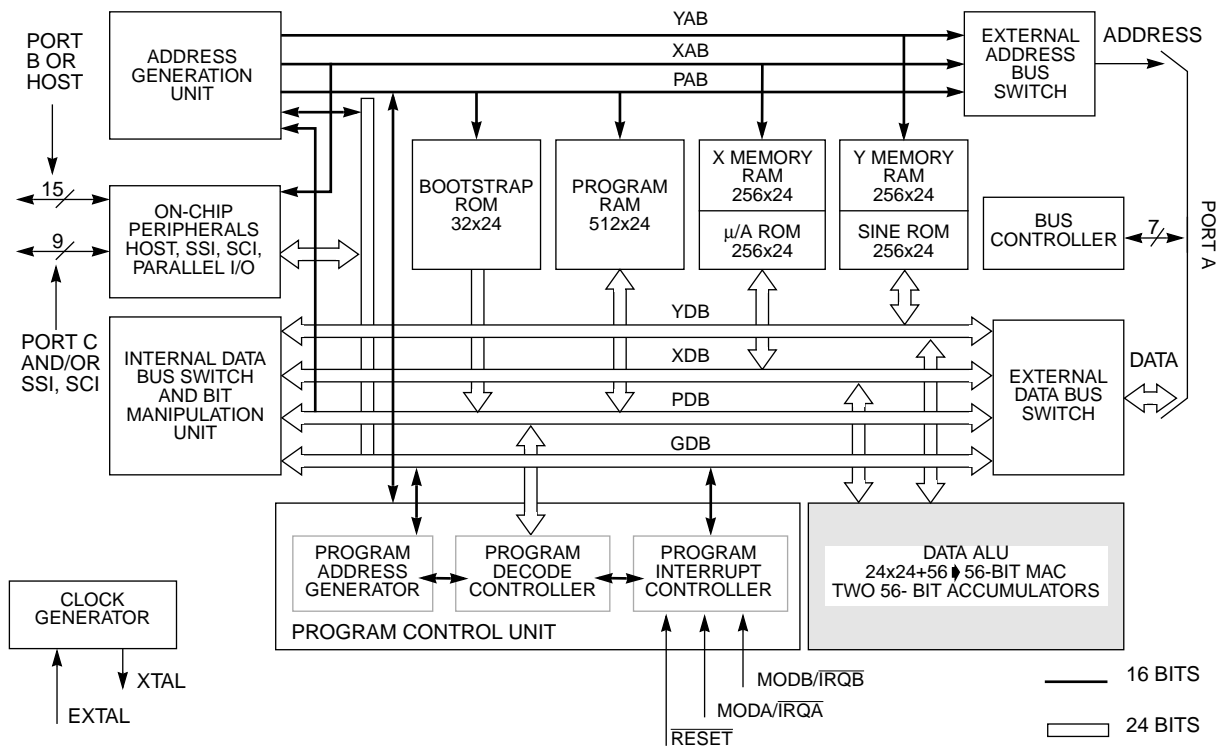


Figure 4-1 DSP56001 Block Diagram

oriented and are designed to interface over the system buses with memory and memory-mapped I/O devices. The DSP56000/DSP56001 instruction set has been designed to allow flexible control of these parallel processing resources. Many instructions allow the programmer to keep each unit busy, thus enhancing performance. It was possible to make the programming model like that of conventional microprocessor units (MPUs), eliminating the need to refer to the detailed chip architecture when programming the DSP56000/DSP56001 because the parallel execution units appear to execute their operations in a nonpipelined manner.

The data ALU (see Figure 4-2r) is the first of these execution units to be presented. The data ALU, which has been designed to be fast and yet provide the capability to process signals having a wide dynamic range, performs all the arithmetic and logical operations on data operands in the DSP56000/DSP56001.

The data ALU registers may be read or written over the XDB and the YDB as 24- or 48-bit operands. The source operands for the data ALU, which may be 24, 48, or 56 bits, always originate from data ALU registers. The results of all data ALU operations are stored in an accumulator.

The 24-bit data words provide 144 dB of dynamic range. This range is sufficient for most real-world applications since the majority of data converters are 16 bits or less, and certainly not greater than 24 bits. The 56-bit accumulator internal to the data ALU provides 336 dB of internal dynamic range so that no loss of precision will occur due to intermediate processing. Circuitry has been provided to facilitate handling data overflows and roundoff errors.

Any of the following operations can be performed by the data ALU in a single instruction cycle: multiplication, multiply-accumulate with positive or negative accumulation, convergent rounding, multiply-accumulate with positive or negative accumulation and convergent rounding, addition, subtraction, a divide iteration, a normalization iteration, shifting, and logical operations.

The components of the data ALU are as follows:

- Four 24-bit input registers
- A parallel, single-cycle, nonpipelined multiply-accumulator/logic unit (MAC)
- Two 48-bit accumulator registers
- Two 8-bit accumulator extension registers
- An accumulator shifter
- Two data bus shifter/limiter circuits

Each of these components is described in the following paragraphs as well as a description of data representation, rounding, and saturation arithmetic.

4.1.1 Data ALU Input Registers (X1, X0, Y1, Y0)

X1, X0, Y1, and Y0 are four 24-bit, general-purpose data registers. They can be treated as four independent, 24-bit registers or as two 48-bit registers called X and Y, developed by the concatenation of X1:X0 and Y1:Y0, respectively. X1 is the most significant word in X and Y1 is the most significant word in Y. The registers serve as input buffer registers between the XDB or YDB and the MAC unit. They are used as data ALU source operands, allowing new operands to be loaded for the next instruction while the register con-

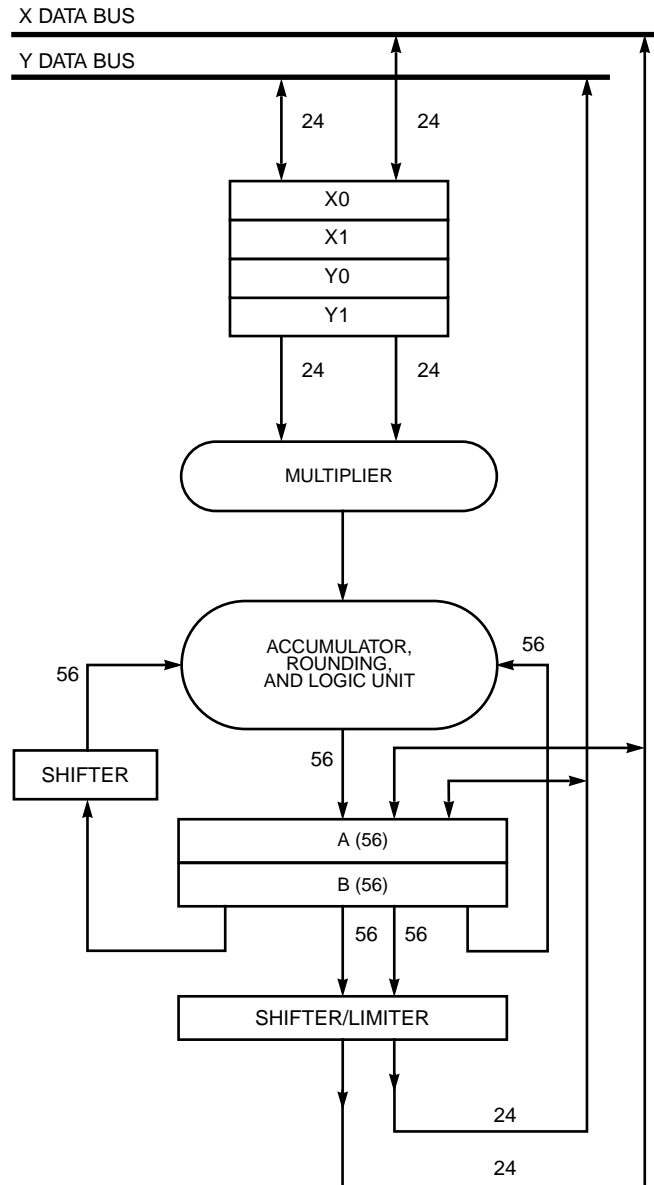


Figure 4-2 Data ALU

tents are used by the current instruction. The registers may also be read back out to the appropriate data bus to implement memory-delay operations and save/restore operations for interrupt service routines.

4.1.2 MAC and Logic Unit

The MAC and logic unit comprise the main arithmetic processing unit of the DSP and perform all of the calculations on data operands. In the case of arithmetic instructions, the unit accepts up to three input operands and outputs one 56-bit result of the following form, extension:most significant product:least significant product (EXT:MSP:LSP). The operation of the MAC unit occurs independently and in parallel with XDB and YDB activity, and its registers facilitate buffering for both data ALU inputs and outputs. Latches are provided on the MAC unit input to permit writing an input register, which is the source for a data ALU operation in the same instruction.

The arithmetic unit contains a multiplier and two accumulators. The input to the multiplier can only come from the X or Y registers (X1, X0, Y1, Y0). The multiplier executes 24-bit x 24-bit, parallel, twos-complement fractional multiplies. The 48-bit product is right justified and added to the 56-bit contents of either the A or B accumulator. The 56-bit sum is stored back in the same accumulator (see Figure 4-3r). An 8-bit adder, which is used as an extension accumulator for the MAC array, accommodates overflow of up to 256 and allows the two 56-bit accumulators to be added and subtracted from each other. The extension adder output is the EXT portion of the MAC unit output. This multiply/accumulate operation is not pipelined but rather is a single-cycle operation. If a multiply without accumulation (MPY) is specified in the instruction, the MAC clears the accumulator and then adds the contents to the product.

In summary, the results of all arithmetic instructions are valid (sign-extended and zero-filled) 56-bit operands in the form of EXT:MSP:LSP or A2:A1:A0 or B2:B1:B0. When a 56-bit result is to be stored as a 24-bit operand, the LSP can be simply truncated, or it can be rounded (using convergent rounding) into the MSP.

Convergent rounding (round-to-nearest) is performed when adding the multiplier's product to the contents of the accumulator if specified in the DSP instruction (e.g., the signed multiply-accumulate and round (MACR) instruction). The bit in the accumulator that is rounded is specified by the scaling mode bits in the status register.

The logic unit performs the logical operations, AND, OR, EOR, and NOT, on data ALU registers. This unit is 24 bits wide and operates on data in the MSP portion of the accumulator. The LSP and EXT portions of the accumulator are not affected.

4.1.3 Data ALU Accumulator Registers (A2, A1, A0, B2, B1, B0)

The six data ALU registers (A2, A1, A0, B2, B1, and B0) form two general-purpose, 56-

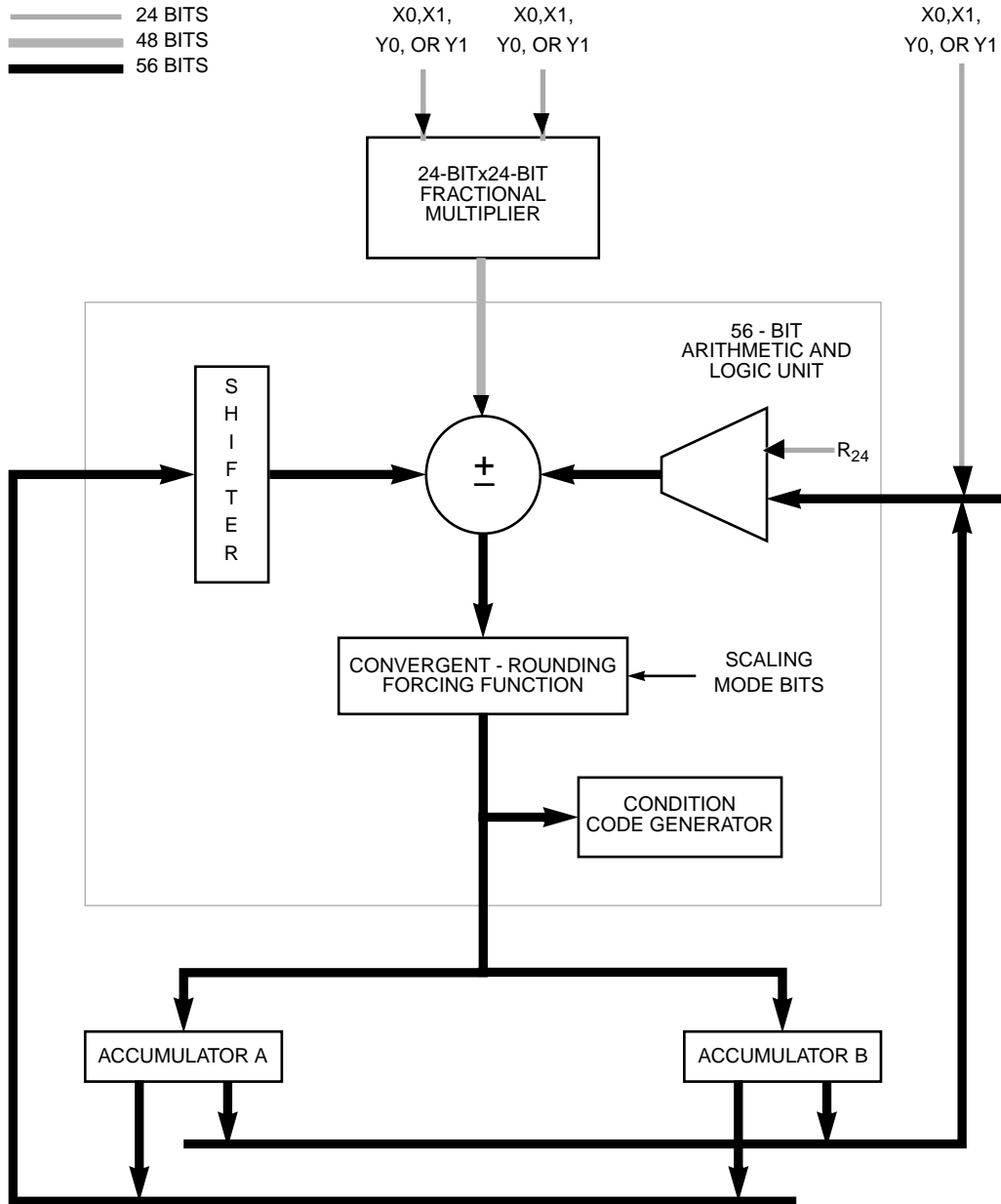


Figure 4-3 MAC Unit

bit accumulators, A and B. Each of these two registers consists of three concatenated registers (A2:A1:A0 and B2:B1:B0, respectively). The 24-bit MSP is stored in A1 or B1; the 24-bit LSP is stored in A0 or B0. The 8-bit EXT is stored in A2 or B2.

The 8-bit extension registers offer protection against overflow. On the DSP56000/DSP56001, the extreme values that a word operand can assume are - 1 and +

0.9999998. If the sum of two numbers is less than - 1 or greater than + 0.9999998, the result (which cannot be represented in a word operand =m i.e., 24 bits) has underflowed or overflowed. The 8-bit extension registers can accurately represent the result of 255 overflows or 255 underflows. Whenever the accumulator extension registers are in use, the V bit in the status register is set.

Automatic sign extension is provided when writing to the 56-bit accumulators A or B with a 48- or 24-bit operand. When a 24-bit operand is written, the low-order portion will be automatically zero filled to form a valid 56-bit operand. The registers may also be written without sign extension or zero fill by specifying the individual register name. When accumulator registers A or B are read, they may be optionally scaled one bit left or one bit right for block floating-point arithmetic.

Reading the A or B accumulators over the XDB and YDB is protected against overflow by substituting a limiting constant for the data that is being transferred. The content of A or B is not affected should limiting occur; only the value transferred over the XDB or YDB is limited. This overflow protection is performed after the contents of the accumulator have been shifted according to the scaling mode. Shifting and limiting will be performed only when the entire 56-bit A or B register is specified as the source for a parallel data move over the XDB or YDB. When A0, A1, A2, B0, B1, or B2 are specified as the source for a parallel data move, shifting and limiting are not performed. The accumulator registers serve as buffer registers between the MAC unit and the XDB and/or YDB. These registers are used as both data ALU source and destination operands.

Automatic sign extension of the 56-bit accumulators is provided when the A or B register is written with a smaller operand. Sign extension can occur when writing A or B from the XDB and/or YDB or with the results of certain data ALU operations (such as the transfer conditionally (Tcc) or transfer data ALU register (TFR) instructions). If a word operand is to be written to an accumulator register (A or B), the MSP (A1 or B1) portion of the accumulator is written with the word operand, the LSP (A0 or B0) portion is zero filled, and the EXT (A2 or B2) portion is sign extended from MSP. Long-word operands are written into the low-order portion, MSP:LSP, of the accumulator register, and the EXT portion is sign extended from MSP. No sign extension is performed if an individual 24-bit register is written (A1, A0, B1, or B0). Test logic is included in each accumulator register to support operation of the data shifter/limiter circuits. This test logic is used to detect overflows out of the data shifter so that the limiter can substitute one of several constants to minimize errors due to the overflow. This process is commonly referred to as saturation arithmetic.

4.1.4 Accumulator Shifter

The accumulator shifter (see Figure 4-3) is an asynchronous parallel shifter with a 56-bit input and a 56-bit output that is implemented immediately before the MAC accumulator

input. The source accumulator shifting operations are as follows:

- No Shift (Unmodified)
- 1-Bit Left Shift (Arithmetic or Logical) ASL, LSL, ROL
- 1-Bit Right Shift (Arithmetic or Logical) ASR, LSR, ROR
- Force to zero

4.1.5 Data Shifter/Limiter

The data shifter/limiter circuits (see Figure 4-3) provide special postprocessing on data read from the ALU accumulator registers A and B out to the XDB or YDB. There are two independent shifter/limiter circuits (one for XDB and one for the YDB); each consists of a shifter followed by a limiting circuit.

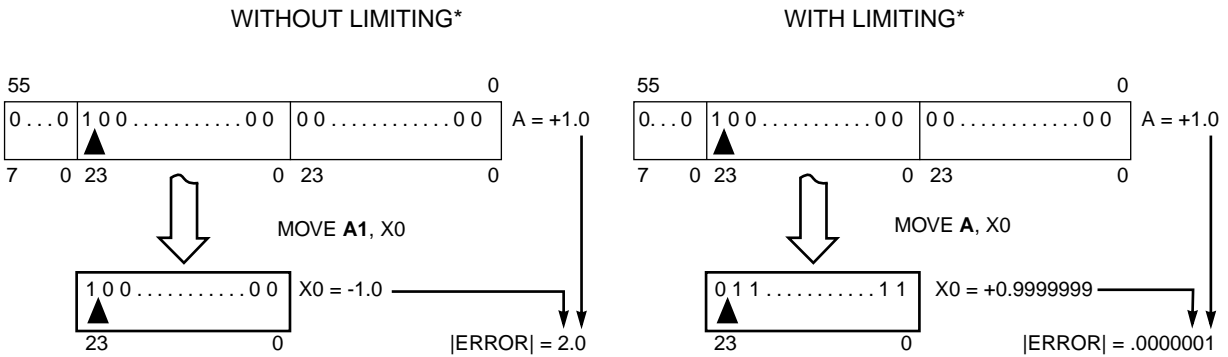
4.1.5.1 Limiting (Saturation Arithmetic)

In the DSP56000/DSP56001, the data ALU accumulators A and B have eight extension bits. Limiting will occur when the extension bits are in use and either A or B is the source being read over XDB or YDB. The limiters in the DSP56000/DSP56001 place a shifted and limited value on XDB or YDB without changing the contents of the A or B registers. Having two limiters allows two-word operands to be limited independently in the same instruction cycle. The two data limiters can also be combined to form one 48-bit data limiter for long-word operands.

If the contents of the selected source accumulator can be represented without overflow in the destination operand size (i.e., accumulator extension register not in use), the data limiter is disabled, and the operand is not modified. If contents of the selected source accumulator cannot be represented without overflow in the destination operand size, the data limiter will substitute a limited data value having maximum magnitude (saturated) and having the same sign as the source accumulator contents: \$7FFFFFFF for 24-bit or \$7FFFFFFF FFFFFFFF for 48-bit positive numbers, \$800000 for 24-bit or \$800000 000000 for 48-bit negative numbers. This process is called saturation arithmetic. The value in the accumulator register is not shifted and can be reused within the data ALU. When limiting does occur, a flag is set and latched in the status register.

For example, if the source operand were 01.100 (+ 1.5 decimal) and the destination register were only four bits, the destination register would contain 1.100 (- 1.5 decimal) after the transfer, assuming signed fractional arithmetic. This is clearly in error as overflow has occurred. To minimize the error due to overflow, it is preferable to write the maximum (“limited”) value the destination can assume. In the example, the limited value would be 0.111 (+ 0.875 decimal), which is clearly closer to + 1.5 than - 1.5 and therefore introduces less error.

Figure 4-4 shows the effects of saturation arithmetic on a move from register A1 to register X0. The instruction “MOVE A1,X0” causes a move without limiting, and the instruc-



* Limiting automatically occurs when the 56 - bit operands A or B (not A2, A1, A0, B2, B1, or B0) are read. The contents of A or B are **NOT** changed.

Figure 4-4 Saturation Arithmetic

tion “MOVE A,X)” causes a move of the same 24 bits with limiting. The error without limiting is 2.0; whereas, it is 0.0000001 with limiting. Table 4-1 shows a more complete set of limiting situations.

4.1.5.2 Scaling

The data shifters are capable of shifting data one bit to the left or one bit to the right as

Table 4-1 Limited Data Values

Destination Memory Reference	Source Operand	Accumulator Sign	Limited Value (Hexadecimal)		Type of Access
			XDB	YDB	
X	X:A	+	7FFFFFFF	—	One 24 bit
	X:B	-	800000	—	
Y	Y:A	+	—	7FFFFFFF	One 24 bit
	Y:B	-	—	800000	
X and Y	X:A Y:A	+	7FFFFFFF	7FFFFFFF	Two 24 bit
	X:A Y:B	-	800000	800000	
	X:B Y:A	+	7FFFFFFF	7FFFFFFF	
	X:B Y:B	-	800000	800000	
	L:AB	+	7FFFFFFF	7FFFFFFF	
	L:BA	-	800000	800000	
L (X:Y)	L:A	+	7FFFFFFF	FFFFFFF	One 48 bit
	L:B	-	800000	000000	

well as passing the data unshifted. Each data shifter has a 24-bit output with overflow indication and is controlled by the scaling mode bits in the status register. These shifters permit dynamic scaling of fixed-point data without modifying the program code. For example, this permits block floating-point algorithms such as fast Fourier transforms to be implemented in a regular fashion.

4.2 DATA REPRESENTATION AND ROUNDING

The DSP56000/DSP56001 uses a fractional data representation for all data ALU operations. Figure 4-5 shows the bit weighting of words, long words, and accumulator oper-

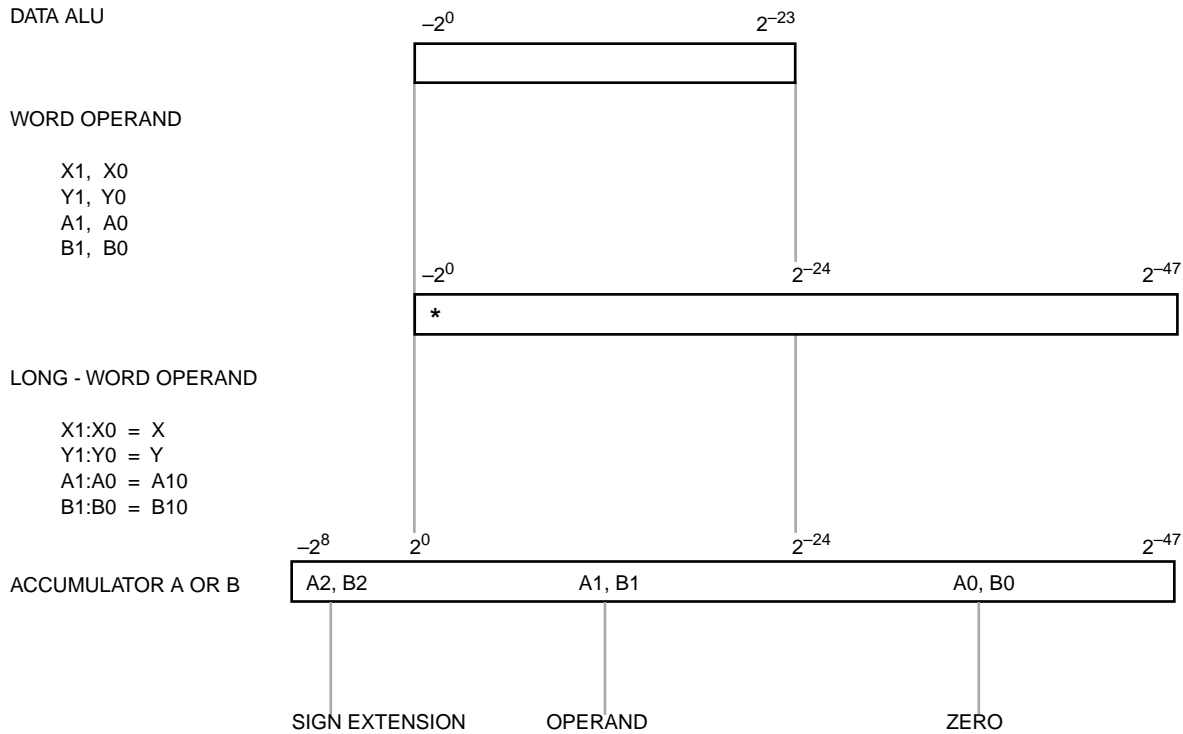


Figure 4-5 Bit Weighting and Alignment of Operands

ands for this representation. The decimal points are all aligned and are left justified.

Data must be converted to a fractional number by scaling before being used by the DSP56000/DSP56001, or the user will have to be very careful in how the DSP manipulates the data. Moving \$3F to a 24-bit data ALU register does not result in the contents being \$00003F as might be expected. Assuming numbers are fractional, the DSP left justifies rather than right justifies. As a result, storing \$3F in a 24-bit register results in the contents being \$3F0000. The simplest example of scaling is to convert all integer num-

bers to fractional numbers by shifting the decimal 24 places to the left (see Figure 4-6).

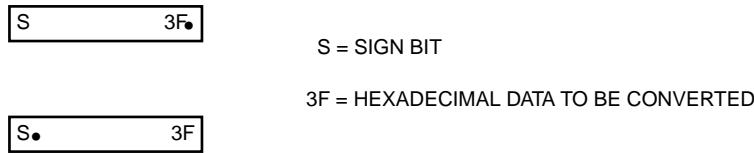


Figure 4-6 Integer-to-Fractional Data Conversion

Thus, the data has not changed; only the position of the decimal has moved.

For words and long words, the most negative number that can be represented is; -1 whose internal representation is \$800000 and \$800000000000, respectively. The most positive word is \$7FFFFFFF or 1 - 2 - 23 and the most positive long word is \$7FFFFFFFFFFFFF or 1 - 2 - 47. These limitations apply to all data stored in memory and to data stored in the data ALU input buffer registers. The extension registers associated with the accumulators allow word growth so that the most positive number that can be used is approximately 256 and the most negative number is approximately; -256. When the accumulator extension registers are in use, the data contained in the accumulators cannot be stored exactly in memory or other registers. In these cases, the data must be limited to the most positive or most negative number consistent with the size of the destination and the sign of the accumulator (the most significant bit (MSB) of the extension register).

To maintain alignment of the binary point when a word operand is written to accumulator A or B, the operand is written to the most significant accumulator register (A1 or B1), and its MSB is automatically sign extended through the accumulator extension register. The least significant accumulator register is automatically cleared. When a long-word operand is written to an accumulator, the least significant word of the operand is written to the least significant accumulator register (see Figure 4-7).

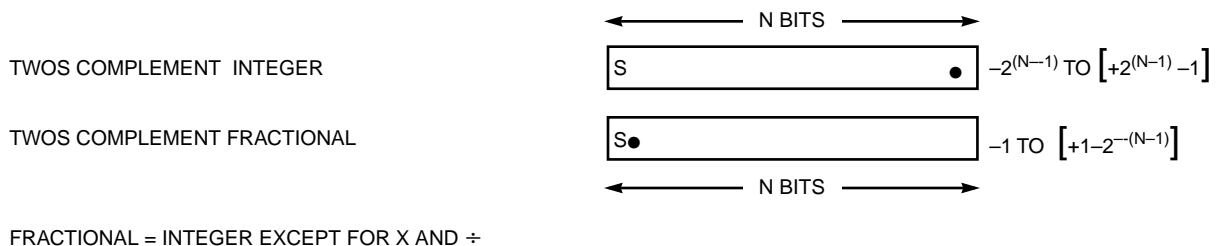


Figure 4-7 Integer/Fractional Number Comparison

A comparison between integer and fractional number representation is shown in Figure 4-7. The number representation for integers is between $\pm 2^{(N-1)}$; whereas, the fractional representation is limited to numbers between ± 1 . To convert from an integer to a fractional number, the integer must be multiplied by a scaling factor so the result will always be between ± 1 . The representation of integer and fractional numbers is the same if the numbers are added or subtracted but is different if the numbers are multiplied or divided. An example of two numbers multiplied together is given in Figure 4-8. The key difference

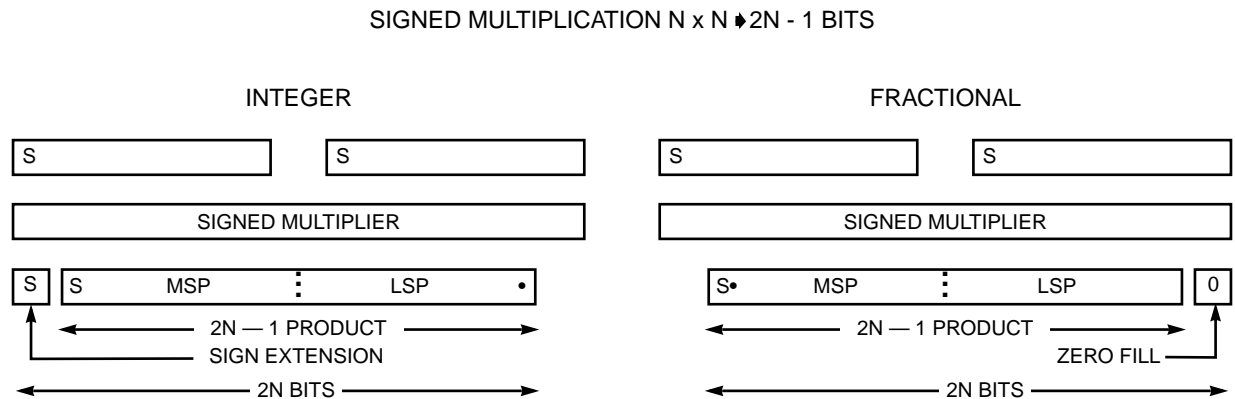


Figure 4-8 Integer/Fractional Multiplication Comparison

is that the extra bit in the integer multiplication is used as a duplicate sign bit and as the least significant bit (LSB) in the fractional multiplication. The advantages of fractional data representation are as follows:

The MSP (left half) has the same format as the input data.

The LSP (right half) can be rounded into the MSP without shifting or updating the exponent.

A significant bit is not lost through sign extension.

Conversion to floating-point representation is easier because the industry-standard floating-point formats use fractional mantissas.

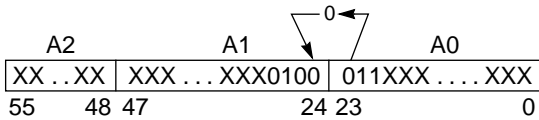
Coefficients for most digital filters are derived as fractions by the high-level language programs used in digital-filter design packages, which implies that the results can be used without the extensive data conversions that other formats require.

Should integer arithmetic be required in an application, shifting a one or zero, depending on the sign, into the MSB converts a fraction to an integer.

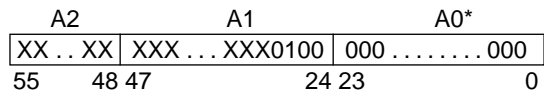
The data ALU MAC performs rounding of the accumulator register to single precision if

CASE I: IF $A0 < \$800000$ (1/2), THEN ROUND DOWN (ADD NOTHING)

BEFORE ROUNDING

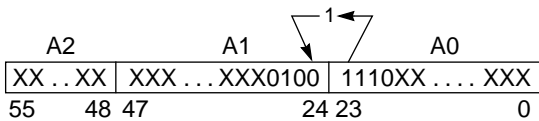


AFTER ROUNDING

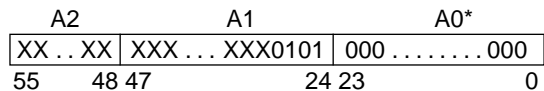


CASE II: IF $A0 > \$800000$ (1/2), THEN ROUND UP (ADD 1 TO A1)

BEFORE ROUNDING

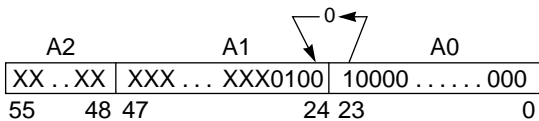


AFTER ROUNDING

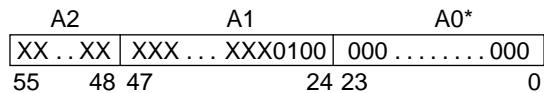


CASE III: IF $A0 = \$800000$ (1/2), AND THE LSB OF A1 = 0, THEN ROUND DOWN (ADD NOTHING)

BEFORE ROUNDING

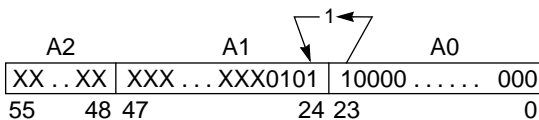


AFTER ROUNDING

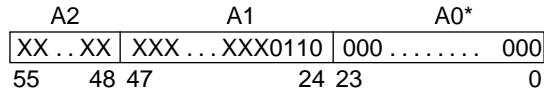


CASE IV: IF $A0 = \$800000$ (1/2), AND THE LSB = 1, THEN ROUND UP (ADD 1 TO A1)

BEFORE ROUNDING



AFTER ROUNDING



*A0 is always clear; performed during RND, MPYR, MACR

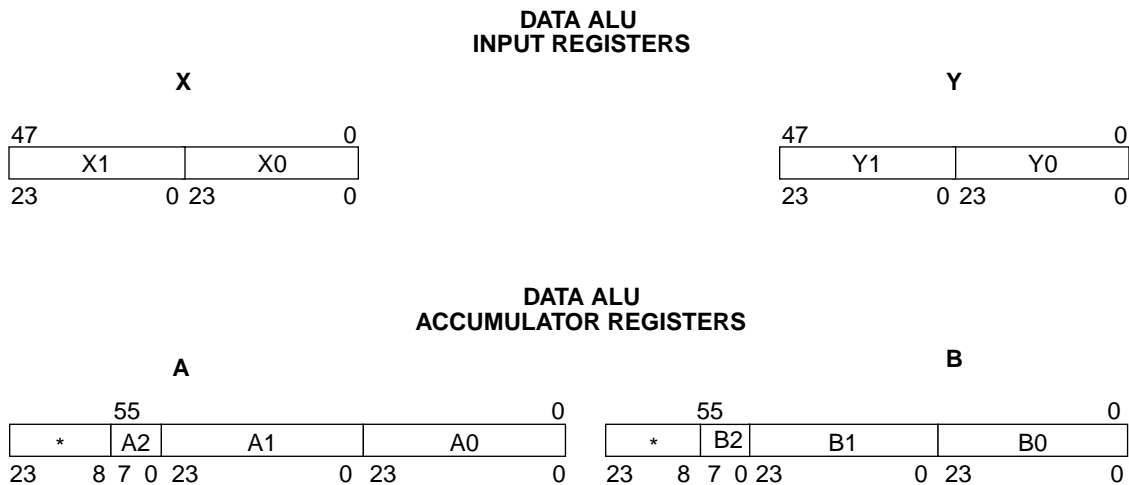
Figure 4-9 Convergent Rounding

requested in the instruction (the A1 or B1 register is rounded according to the contents of the A0 or B0 register). The rounding method used is called round-to-nearest (even) number, sometimes referred to as convergent rounding. The usual rounding method rounds up any value above one-half and rounds down any value below one-half. The question arises as to which way one-half should be rounded. If it is always rounded one way, the

results will eventually be a bias in that direction. Convergent rounding solves the problem by rounding down if the number is odd (LSB=0) and rounding up if the number is even (LSB=1). Figure 4-9 shows the four cases for rounding a number in the A1 (or B1) register. If scaling is set in the status register, the resultant number will be rounded as it is put on the data bus. However, the contents of the register are not scaled.

4.3 DATA ALU PROGRAMMING MODEL

The data ALU features 24-bit input/output data registers that can be concatenated to accommodate 48-bit data and two 56-bit accumulators, which are segmented into three 24-bit pieces that can be transferred over the buses. Figure 4-10 illustrates how the registers in the programming model are grouped.



*Read as sign extension bits, written as don't care.

Figure 4-10 DSP56000/DSP56001 Programming Model

4.4 DATA ALU SUMMARY

The data ALU is optimized for arithmetic operations involving multiply and accumulate operations with two separate data spaces. The data ALU, which executes all instructions in one machine cycle, is not pipelined. The two 24-bit numbers being multiplied can come from the X registers (X0 or X1) or Y registers (Y0 or Y1). After multiplication, they are added (or subtracted) with one of the 56-bit accumulators and can be convergently rounded to 24 bits. The convergent-rounding forcing function detects the \$800000 condition in the LSP and makes the correction as necessary. The final result is then stored in one of the accumulators as a valid 56-bit number. The condition code bits are set based on the rounded output of the logic unit.

