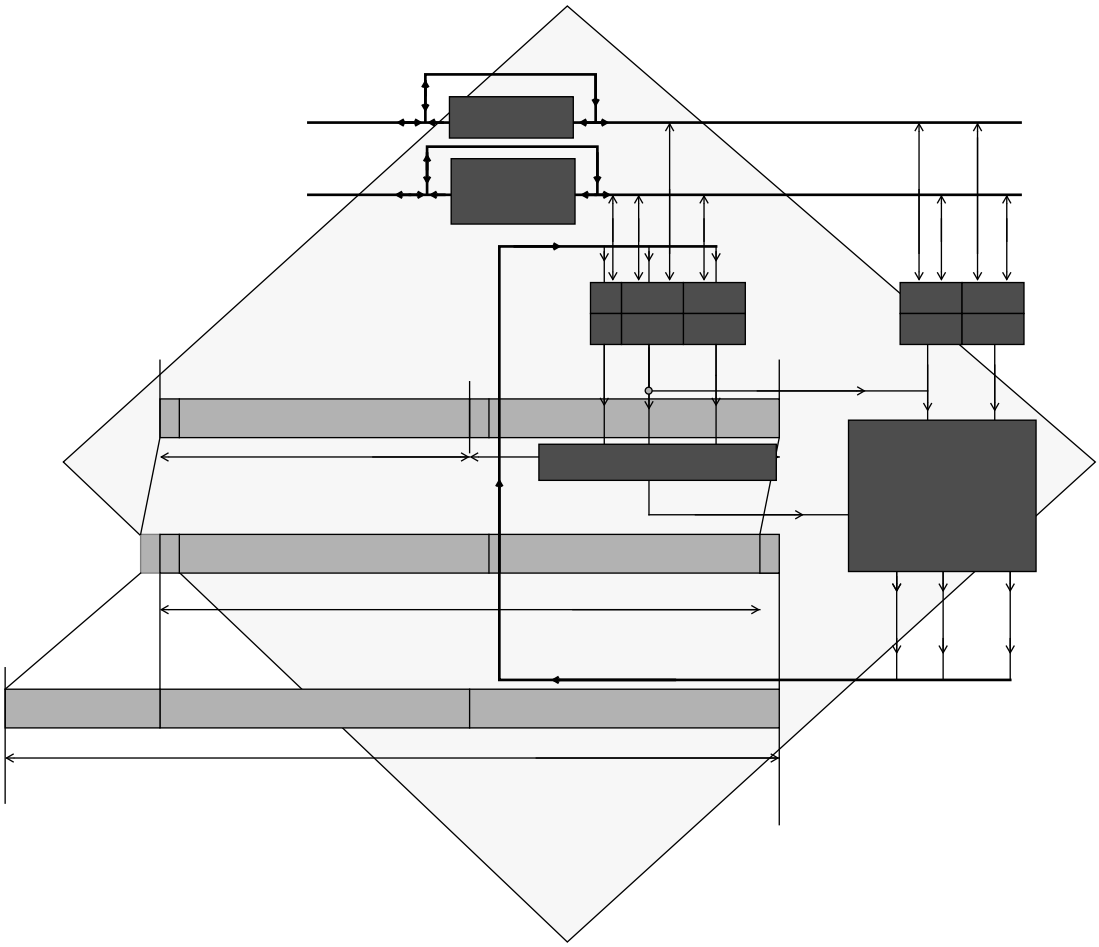
SECTION 3

DATA ALU



SECTION CONTENTS

3.1 OVERVIEW AND ARCHITECTURE

This Section describes the structure and the operation of the Data ALU registers and hardware in addition to describing the data representation, rounding, and saturation arithmetic used within the Data ALU.

The major components of the Data ALU are

- Data ALU Input Registers
- Data ALU Accumulator Registers
- A parallel single cycle non-pipelined Multiply-Accumulator (MAC) Unit
- An Accumulator Shifter (AS)
- An Output Shifter (OS)
- A Data Shifter/Limiter (S/L)

A block diagram of the Data ALU architecture is shown in Figure 3-1 and a functional block diagram is shown in Figure 3-2.

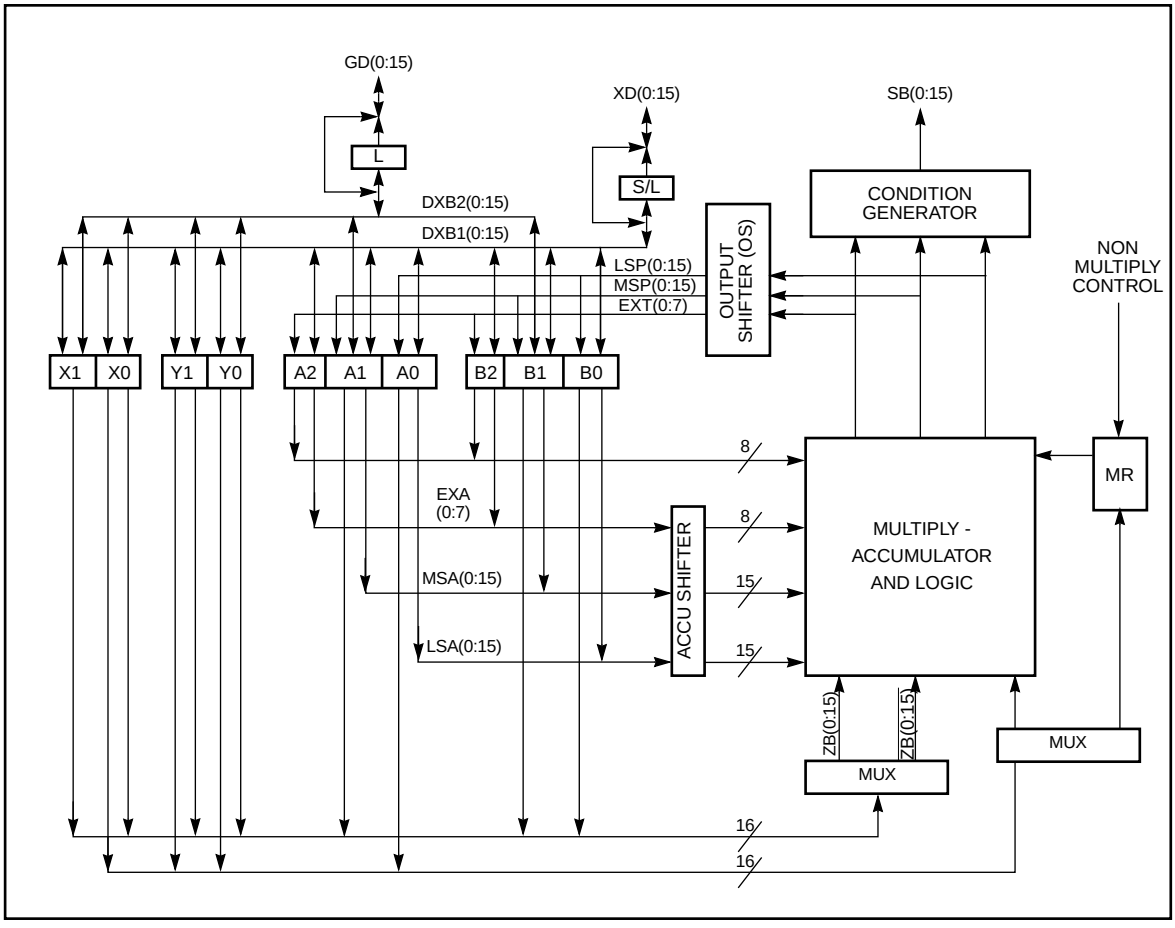


Figure 3-1 Data ALU Architecture Block Diagram

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

X1, X0, Y1, and Y0 are 16-bit latches which serve as input registers for the data ALU. Each register may be read or written by the XDB as well as the GDB. X0, X1, Y0, and Y1 may be read over the XDB. They may be treated as four independent 16-bit registers or as two 32-bit registers called X and Y which are developed by concatenating X1:X0 and Y1:Y0 respectively (where X1 and Y1 are the most significant words and X0 and Y0 are the least significant words in X and Y respectively).

These Data ALU input registers are used as source operands for most data ALU operations and allow new operands to be loaded for the next instruction while the register contents are used by the current instruction.

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

A1, A0, B1 and B0 are 16-bit latches which serve as data ALU accumulator registers. A2 and B2 are 8-bit latches which serve as accumulator extension registers. Each register may be read or written by the XDB as a word operand. A1 and B1 may be read or written by the GDB. When A2 or B2 is read, the register contents occupy the low-order portion (bits 7-0) of the word; the high-order portion (bits 16-8) is sign-extended. When A2 or B2 is written, the register receives the low-order portion of the word; the high-order portion is not used.

The accumulator registers are treated as two 40-bit registers A (A2:A1:A0) and B (B2:B1:B0) for data ALU operations. These accumulator registers receive the EXT:MSP:LSP portion of the Multiply-Accumulator unit output and supply a source accumulator of the same form. Most data ALU operations specify the 40-bit accumulator registers as source and/or destination operands

The accumulator registers are treated as two 40-bit registers A (A2:A1:A0) and B (B2:B1:B0) for data ALU operations. These accumulator registers receive the EXT:MSP:LSP portion of the Multiply-Accumulator unit output and supply a source accumulator of the same form. Most data ALU operations specify the 40-bit accumulator registers as source and/or destination operands.

When one accumulator is used as a multiplier input, only the upper portion (A1 or B1) can be specified. This upper portion can also be directly used as an address register for fast effective address computation.

Automatic sign extension of the 40-bit accumulators is provided when the A or B register is written with a smaller size operand. This can occur when writing A or B from the X data bus or with the results of certain data ALU operations (such as Tcc or TFR). If a word operand is to be written to an accumulator register (A or B), the MSP portion of the accu-

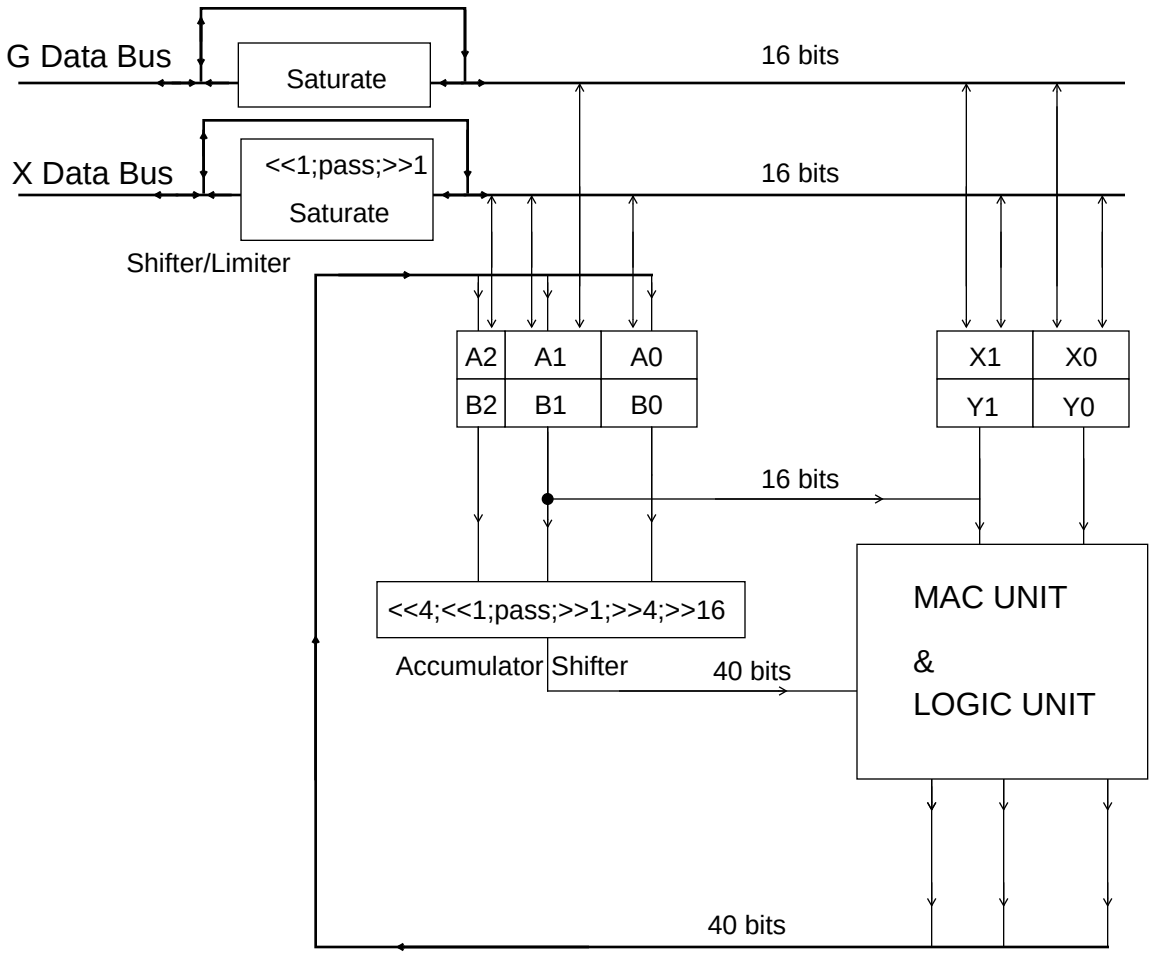


Figure 3-2 Data ALU Functional Block Diagram

mulator is written with the word operand, the LSP portion is zeroed and the EXT portion is sign-extended from MSP. No sign extension is performed if an individual 16-bit register (A1, A0, B1, or B0) is written.

The extension registers A2 and B2 offer protection against 32-bit overflow. When the result of an accumulation crosses the MSB of MSP (bit 15 of A1 or B1), the extension bit of the status register (E bit) is set. Up to 255 overflows or underflows are possible using this extension byte, after which the sign is lost beyond the MSB of the EXT register, setting the overflow bit (V bit) in the status register.

It is also possible to saturate the accumulator on a 32-bit value automatically after every accumulation. This is done by setting the saturation bit in the Operating Mode Register (OMR). The highest dynamic range of the machine is limited to 32 bits then, and the limiting bit (L bit) in the status register is set by the saturation.

The detection of the overflow logic is also used to saturate an accumulator out of the shifter/limiter register while reading A or B accumulators over the XDB or transferring them to any data ALU register. The content of A or B is not affected in that case (except when the same accumulator is specified as source and destination); only the value transferred over the XDB is limited to a full-scale positive or negative 16-bit value ($7FFF or $8000), respectively. This overflow protection is performed after the contents of the accumulator have been shifted according to the scaling mode defined in the status register. When limiting occurs, the L bit flag in the status register is set and latched. Note that only when an entire 40 bit accumulator register (A or B) is specified as the source for a parallel data move over the XDB will shifting and limiting be performed. Shifting and limiting are not performed when A0, A1, A2, B0, B1, or B2 are individually specified.

3.1.3 Multiply-Accumulator (MAC) and Logic Unit

The MAC and logic unit is the main arithmetic processing unit of the DSP and performs all of the calculations on data operands. The MAC unit accepts up to three input operands and outputs one 40-bit result of the form Extension:Most Significant Product: Least Significant Product (EXT:MSP:LSP). The operation of the MAC unit occurs independently and in parallel with XDB, GDB, and PDB activity. The Data ALU registers provide pipelining 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. All ALU operations occur in one instruction cycle. The inputs of the multiplier can come from the X and Y registers (X1, X0, Y1, Y0) as well as from the MSP of each accumulator (A1, B1). The multiplier executes 16 x 16-bit parallel signed/unsigned fractional and signed integer multiplies.

For fractional arithmetic, the 31-bit product is added to the 40-bit contents of either the A or B accumulator. The 40-bit sum is stored back in the same accumulator. This multiply/accumulate is a single cycle operation (no pipeline). Integer operations always generate a 16-bit result located in the accumulator MSP portion (A1 or B1). Full precision integer operations are possible using an ASR instruction after any fractional MPY or MAC.

If a multiply without accumulation is specified in the instruction, the MAC clears the accumulator and then adds the contents to the product. The results of all arithmetic instructions are valid (sign extended and zero filled) 40-bit operands in the form EXT:MSP:LSP, A2:A1:A0, or B2:B1:B0 (except during integer operations). When a 40-bit result is to be stored as a 16-bit operand, the LSP can simply be truncated or it can be rounded into the MSP. The rounding performed is either convergent rounding (Round to the nearest even) or twos-complement rounding. The type of rounding is specified by the rounding bit in the status register. The bit in the accumulator which is rounded is specified by the scaling mode bits in the status register.

The major components of the MAC unit are

- Multiply-Accumulator Array
- ZB Multiplexer
- Multiplier Control Recoder
- Extension Adder
- Logic unit

3.1.3.1 Multiply-Accumulator (MAC) Array and Logic unit

The multiply-accumulator array is a 16 X 16-bit asynchronous, parallel multiply-accumulator with 40-bit accumulation. The MAC array is based on the modified Booth's algorithm. The MAC array is used in all arithmetic operations. The array performs signed and unsigned arithmetic with a fractional data representation and signed arithmetic with an integer data representation. The MAC array also performs rounding if specified in the DSP instruction. The type of rounding is specified by the scaling mode bits and the rounding bit in the status register.

Three input operands are received on six internal data buses AS2, AS1, AS0, EB, ZB, and MB. The AS2:AS1:AS0 data bus is the 40-bit source accumulator bus and represents the EXT:MSP:LSP portion of the source accumulator. The AS2:AS1:AS0 bus is the output of the accumulator shifter. The ZB data bus is a 16-bit input operand used in most data ALU operations and represents the multiplicand in multiplication operations. The MB data bus is a 16-bit input operand which represents the multiplier in multiplication operations. The ZB and MB buses are concatenated (ZB:MB) to form a 32-bit input bus for long word operands. The EB bus is concatenated with the ZB and MB buses (EB:ZB:MB) to form a 40-bit input bus for addition or subtraction of the two full accumulators.

The logic unit in the MAC array performs the logical operations AND, OR, EOR, and NOT on data ALU registers. The logic unit is 16 bits wide and operates on data in the MSP portion of the accumulator. The LSP and EXT portions of the accumulator are not affected.

3.1.3.2 ZB Multiplexer

The ZB Multiplexer sign extends, by one bit, the data coming into the MAC over the ZB bus. This sign bit can be cleared by the ZB Multiplexer to obtain an unsigned format for these operands. The ZB Multiplexer may also invert data coming into the MAC as required.

3.1.3.3 Multiplier Control Recoder (REC)

The multiplier control recoder directs the operation of the MAC array and performs multiplier operand recoding for the modified Booth's algorithm multiplication. The MB bus is the input to the multiplier control recoder. Data-independent multiplier control line generation is performed in the REC for most non-multiplication instructions. For example, the multiplier control output for a data ALU addition would be a multiplication by +1 operation. For other data ALU operations, the multiplier control recoder generates control line constants that do not correspond to a valid multiplier control word. The least significant recoder outputs a zero control word and the most significant recoder provides all the functions in these cases.

3.1.3.4 Extension Adder (EXA)

EXA is an 8-bit adder which serves as an extension accumulator for the MAC array. The primary source operand is the AS2 internal data bus from the accumulator shifter. For multiply-accumulate operations, the second source operand is an update constant generated from the carry and overflow outputs of the MAC array. For 40-bit additions or subtractions, the EB internal data bus is used as the second source operand. This allows the two accumulators to be added and subtracted from each other. The extension adder output is the EXT portion of the MAC unit output and is the sum of the source operands.

3.1.4 Accumulator Shifter (AS)

The accumulator shifter is an asynchronous parallel shifter with a 40-bit input and a 40-bit output. The source accumulator shifting operations are:

1. No Shift (Unmodified)
2. 1-Bit Left Shift (Arithmetic) ASL
3. 1-Bit Right Shift (Arithmetic) ASR
4. 4-Bit Right Shift (Arithmetic) ASR4
5. 4-Bit Left Shift (Arithmetic) ASL4
6. 16-Bit Right Shift (Arithmetic) ASR16
7. Force to zero

The shifter also performs a 15-bit arithmetic shift to the right during integer multiply-accumulate (IMAC) instructions. The shifter is implemented immediately before MAC accumulator input. The accumulator shifter output can be inverted or forced to zero and linkages are provided to shift into and out of the condition code carry (C) bit. The accumulator shifter outputs to the AS2, AS1, and AS0 buses in the internal ALU.

3.1.5 Output Shifter (OS)

The Output shifter is an asynchronous parallel shifter with 40-bit input and a 40-bit output. This shifter operates a 15-bit left shift on the result of the integer operations IMPY/IMAC before storing the shifters result into an accumulator. The shifted result is then available in the A1 or B1 MSP for other arithmetic or logical operations.

3.1.6 Data Shifter/Limiter

The data shifter/limiter provides special post processing on data ALU accumulator registers when they are read out to the XDB or to other registers. It consists of a shifter followed by a limiting circuit.

3.1.6.1 Scaling

The data shifter is capable of shifting data one bit to the left or right as well as passing the data unshifted. It has a 16-bit output and a limiting output indicator. The data shifter is controlled by the scaling mode bits in the status register. These mode bits permit dynamic scaling of fixed point data using the same program code which permits block floating point algorithms to be implemented in a regular fashion. FFT routines would typically use this feature to selectively scale each butterfly pass.

3.1.6.2 Limiting

Saturation arithmetic is provided to selectively limit overflow when reading a data ALU accumulator register. Limiting is performed on the data shifter output. If the contents of the selected source accumulator can be represented in the destination operand size without overflow, the data limiter is disabled and the operand is not modified. If the 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 and the same sign as the source accumulator. The value of the accumulator is not changed. The limited data values are shown in Table 3-1

Table 3-1 Saturation by the Shifter/limiter

E bit	MSB of A2/B2	Output of the limiter
0	x	unchanged
1	0	$7FFF
1	1	$8000

The E bit is the extension bit of the status register (SR) which is defined Section 5.3.6. Note that during the TFR2 instruction, the limiting is performed on 32 bits when the accumulator is written to a register.

3.2 THE DATA ALU ARITHMETIC AND ROUNDING

The DSP56100 family supports the two's-complement representation of binary numbers. In this format, the sign bit is the MSB of the binary word, which is set to zero for positive numbers and set to one for negative numbers. Unsigned numbers are only supported by instructions dedicated to multiple precision.

3.2.1 Data Representation

Three modes of format adjustments are supported by the 16-bit DSP:

1. **Two's complement fractional.** In this format, the N bit operand is represented using the 1.[N-1] format (1 signed bit, N-1 fractional bits). Such a format can represent numbers between -1 and $+1-2^{-[N-1]}$
2. **Unsigned fractional.** Unsigned binary numbers may be thought of as positive only. The unsigned numbers have nearly twice the magnitude of a signed number of the same length. An unsigned fraction, D, is a number whose magnitude satisfies the inequality:

   $0.0 \leq D < 2.0$

   Examples of unsigned fractional numbers are 0.25, 1.25, and 1.999. The binary word is interpreted as having a binary point after the most significant bit (MSB). The most positive number is $FFFF or $\{1.0 + (1 - 2^{-[N-1]})\} = 1.99996948$ (for N=16 bits). The smallest positive number is zero ($0000).
3. **Two's complement integer.** This format is used by two instructions, the integer multiply and multiply-accumulate (IMPY/IMAC). Using this format, the N-bit operand is represented using the N.0 format (N integer bits). Such a format can represent numbers between $-2^{-[N-1]}$ and $[2^{[N-1]}-1]$.

The operand is written to the most significant accumulator register (A1 or B1) and its most significant bit is automatically sign extended through the accumulator extension register to maintain alignments of the binary point when a word operand is written to A or B. The least significant accumulator register is automatically cleared. See Figure 3-3 for more details on bit weighting and operand alignments

.

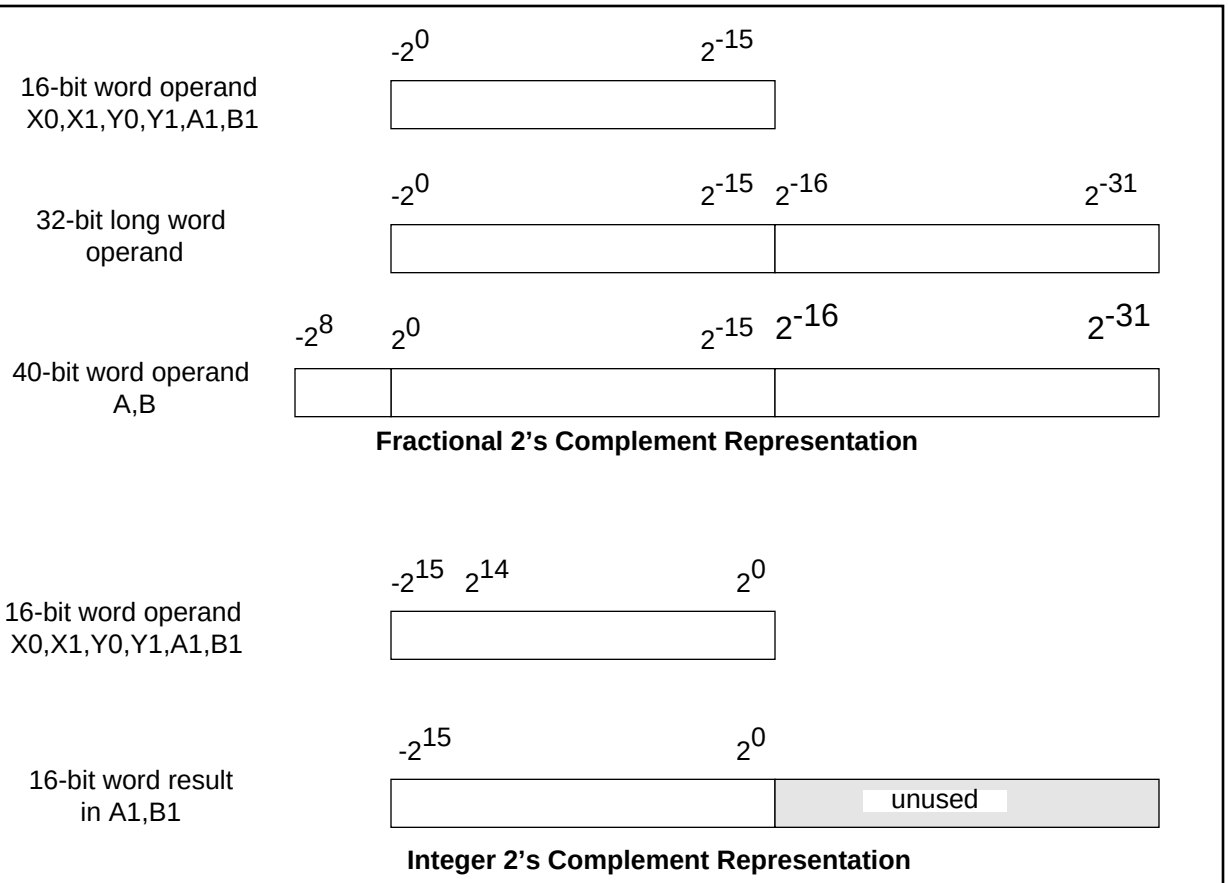


Figure 3-3
Bit Weighting and Alignments for Operands in Fractional and Integer Representation

3.2.2 Fractional Arithmetic

Figure 3-4 shows the Multiply-Accumulation implementation for fractional arithmetic. The multiplication of two 16-bit signed fractional operands gives a 32-bit signed fractional intermediate result with the LSB always set to zero. This intermediate result is added to one of the 40-bit accumulators. If rounding is specified in the MPY or MAC instruction (MACR or MPYR), the intermediate result will be rounded to 16 bits before being stored back to the destination accumulator

.

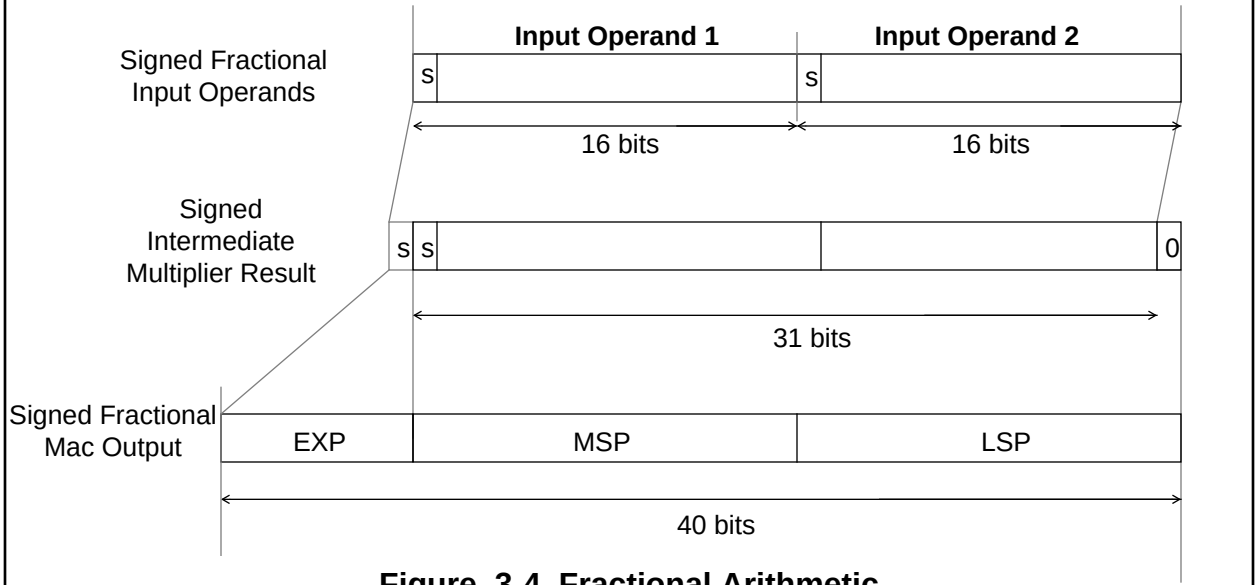


Figure 3-4 Fractional Arithmetic

3.2.3 Integer Arithmetic

Figure 3-5 shows the Multiply and Multiply-Accumulate operations for integer arithmetic and Figure 3-6 describes the implementation of the Integer Multiply-Accumulate. The multiplication/multiply-accumulate of two 16-bit signed integer operands (IMPY/IMAC) gives a 16-bit signed integer result in the MSP (A1 or B1). EXT (A2 or B2) is sign extended and the LSP (A0 or B0) is unchanged. Since A0 and B0 remain unchanged by integer arithmetic instructions, these two registers can be used as two additional data ALU registers when using IMAC, IMPY, INC24, DEC24, CLR24, SWAP, and EXT instructions. Full precision 40-bit integer operations are possible using a fractional MPY or a series of MACs followed by an ASR instruction.

CAUTION

Overflow control and rounding are **not** performed during integer multiplication and integer multiply-accumulate.

Integer arithmetic is optimized for new address generation using the multiplier. For example, when an address register Rn has to be updated to Rn + x0*y0 before fetching new data from memory, the following sequence of code can be used:

```
move    Rn,a            ;a=Rn
imac    x0,y0,a         ;a1=Rn+x0*y0
move    x:(a1),b        ;b1=X:<Rn+x0*y0>
```

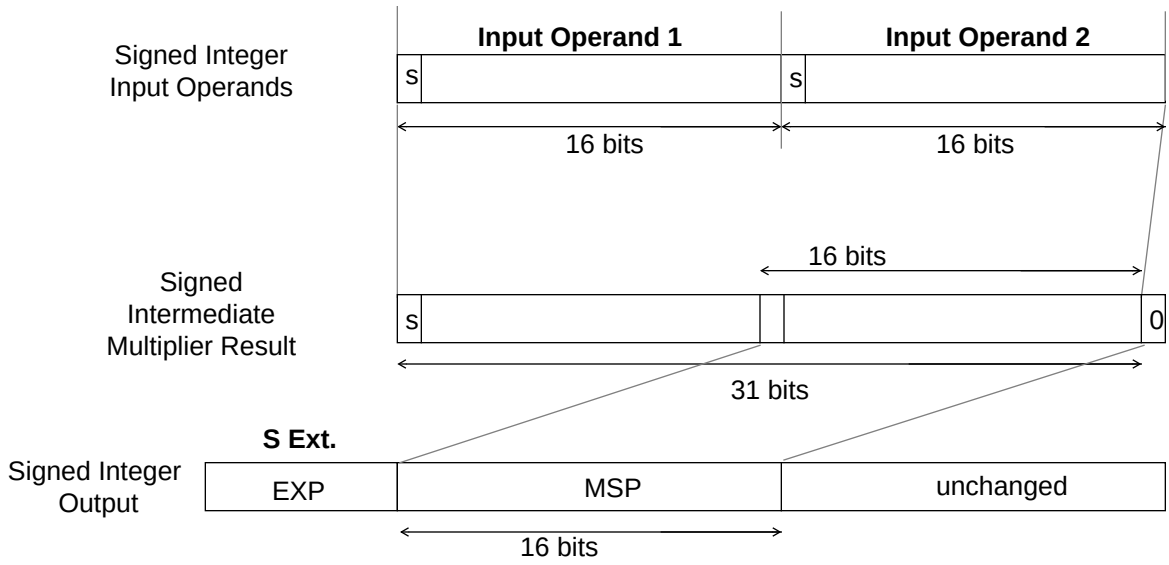


Figure 3-5 Integer Arithmetic (IMPY/IMAC)

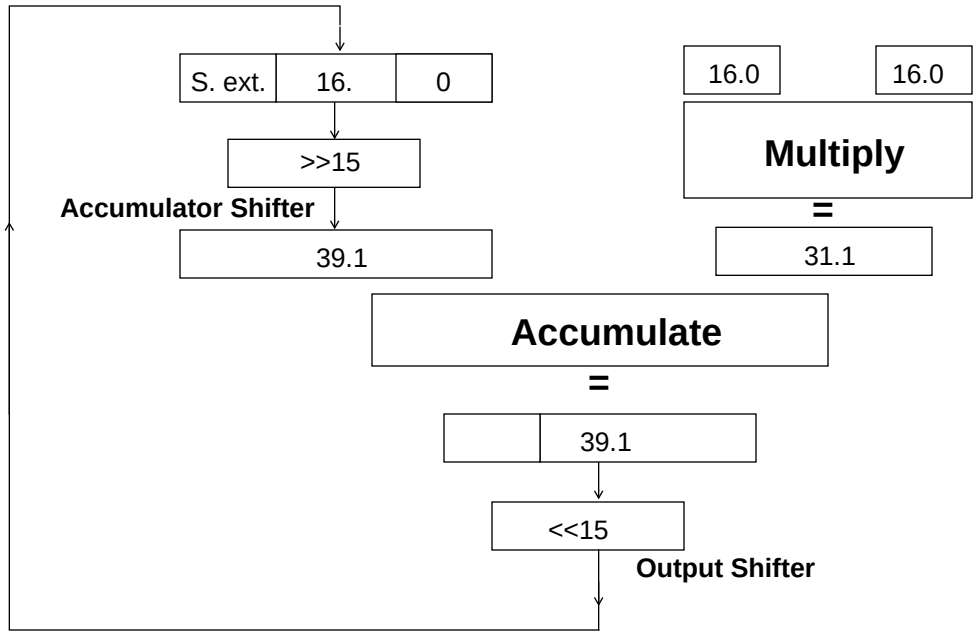


Figure 3-6 IMAC Implementation

3.2.4 Multiprecision Arithmetic Support

A set of data ALU operations is provided in order to facilitate multi-precision multiplications. When these instructions are used, the multiplier accepts some combinations of signed twos-complement format and unsigned format. These instructions are:

1. **MPY/MAC su:** multiplication and multiply-accumulate with signed times unsigned operands
2. **MPY/MAC uu:** multiplication and multiply-accumulate with unsigned times unsigned operands
3. **DMACss:** multiplication with signed times signed operands and 16-bit arithmetic right shift of the accumulator before accumulation
4. **DMACsu:** multiplication with signed times unsigned operands and 16-bit arithmetic right shift of the accumulator before accumulation
5. **DMACuu:** multiplication with unsigned times unsigned operands and 16-bit arithmetic right shift of the accumulator before accumulation

Figure 3-7 shows how the DMAC instruction is implemented inside the Data ALU and Figure 3-8 illustrates the use of these instructions in the case of a double precision multiplication. The signed x signed operation is used to multiply or multiply-accumulate the two upper, signed, portions of two signed double precision numbers. The unsigned x signed operation is used to multiply or multiply-accumulate the upper, signed, portion of one double precision number with the lower, unsigned, portion of the other double precision number. The unsigned x unsigned operation is used to multiply or multiply-accumulate the lower, unsigned, portion of one double precision number with the lower, unsigned, portion of the other double precision number.

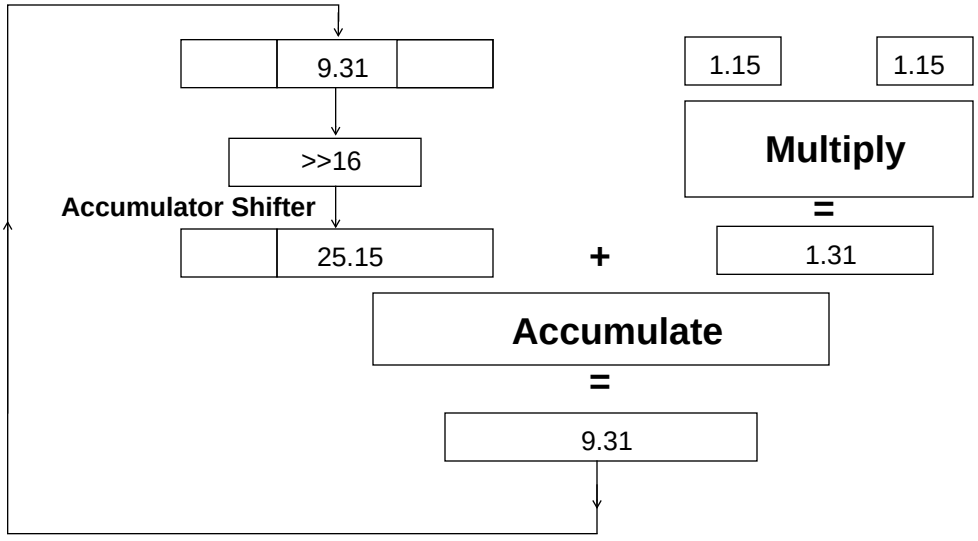


Figure 3-7 DMAC Implementation

3.2.5 Rounding Modes

The DSP56100 family implements two types of rounding: convergent rounding and two's complement rounding. The type of rounding is selected by the OMR rounding bit (R bit).

3.2.5.1 Convergent Rounding

This is the default rounding mode. Convergent rounding is also called round-to-nearest even number. It prevents the introduction of a bias normally produced by rounding down if the number is odd (LSB=1) and rounding up if the number is even (LSB=0). Figure 3-9 shows the four possible cases for rounding a number in the A1 or B1 register. If the Least Significant Portion (LSP) of a number is less than half ($<8000) of the bit to be rounded (LSB), the number is rounded down and if the LSP of the number is greater than half of the LSB (>$8000) the number is rounded up. If the LSP is exactly equal to half of the LSB ($8000) and the LSB of the MSP is odd, the number is rounded up whereas if the LSB of the MSP is even, the number is rounded down i.e., truncated. This technique eliminates the bias in truncation rounding.

Block diagrams of the rounding implementations for the cases of no scaling, scaling down and scaling up are shown in Figure 3-9, Figure 3-10, and Figure 3-11, respectively. Scaling modes require that the zero detect hardware and LSB Even gate have one of three forms since the LSB moves with the scaling mode.

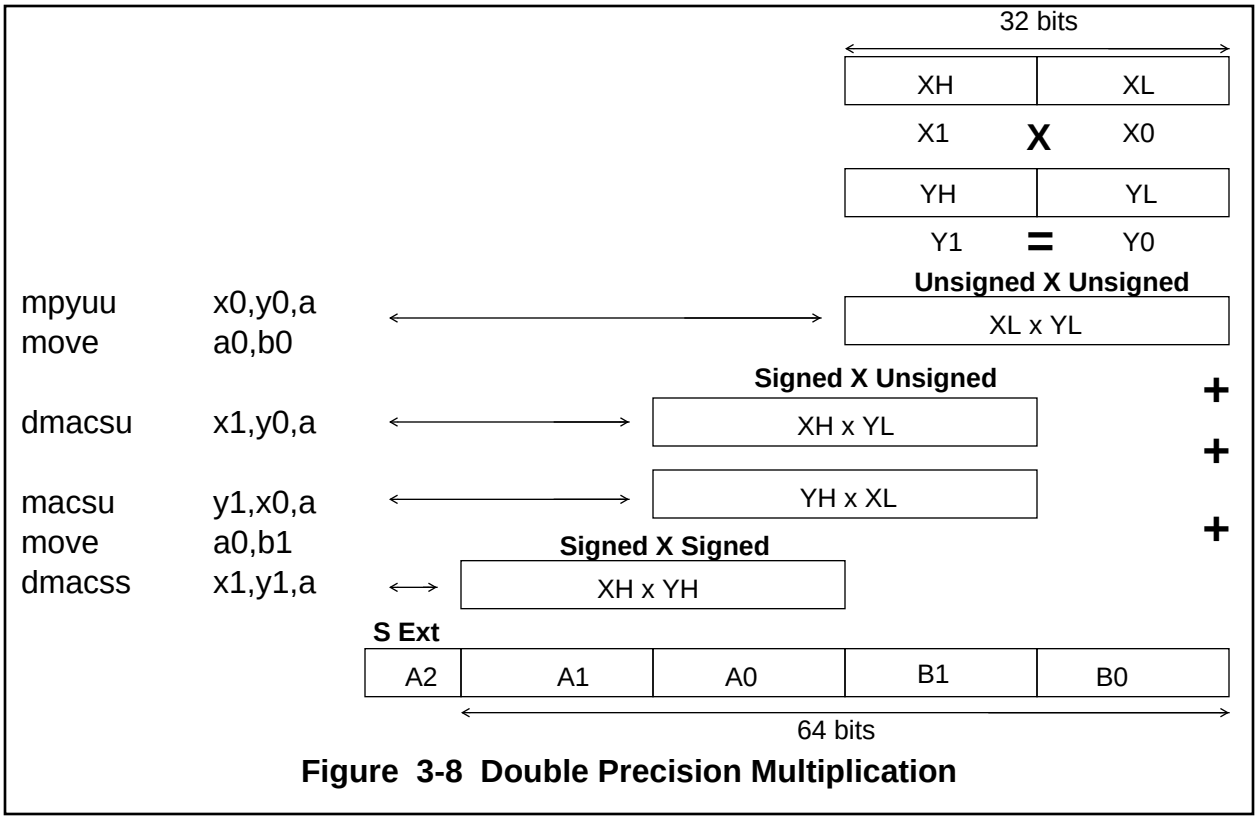


Figure 3-8 Double Precision Multiplication

CASE I: **A0<0.5 (<$8000), then round down (add zero and A1)**

Before Rounding

A2	A1	A0
XX..X	XX...XX0100	011XXX...XX
39	31	15 0

After Rounding

A2	A1	A0
XX..X	XX...XX0100	0000...0000
39	31	15 0

CASE II: **A0>0.5 (>$8000), then round up (add 1 to A1)**

Before Rounding

A2	A1	A0
XX..X	XX...XX0100	1110XX...XX
39	31	15 0

After Rounding

A2	A1	A0
XX..X	XX...XX0101	0000...0000
39	31	15 0

CASE III: **A0=0.5 (=$8000) and LSB of A1=0 (even), then round down (add zero to A1)**

Before Rounding

A2	A1	A0
XX..X	XX...XX0100	1000...0000
39	31	15 0

After Rounding

A2	A1	A0
XX..X	XX...XX0100	0000...0000
39	31	15 0

CASE IV: **A0=0.5 (=$8000) and LSB of A1=1(odd), then round up (add 1 to A1)**

Before Rounding

A2	A1	A0
XX..X	XX...XX0101	1000...0000
39	31	15 0

After Rounding

A2	A1	A0
XX..X	XX...XX0110	0000...0000
39	31	15 0

Figure 3-9 Convergent Rounding

.

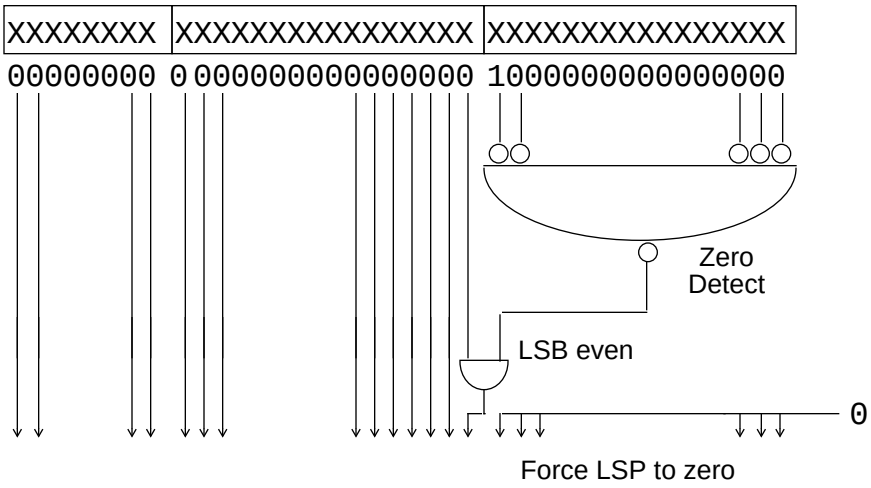


Figure 3-10 Convergent Rounding Implementation – No Scaling

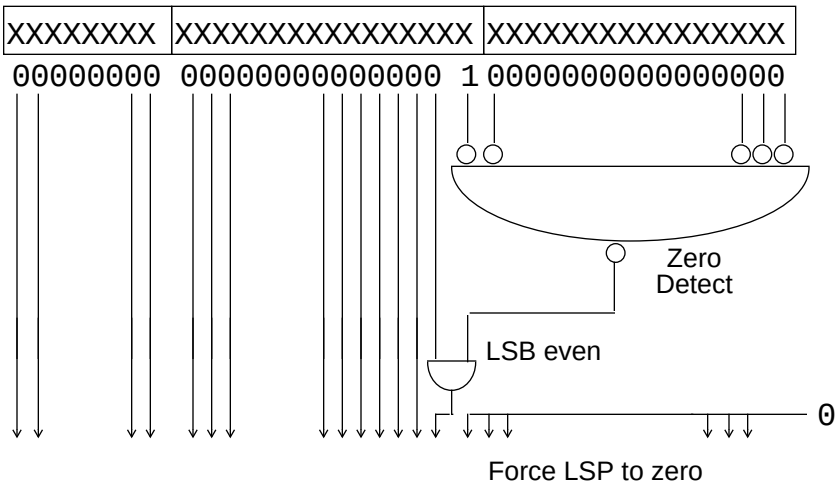


Figure 3-11 Convergent Rounding Implementation – Scale Down

3.2.5.2 Two's Complement Rounding

When twos-complement rounding is selected by setting the rounding bit in the OMR, one is added to the bit to the right of the rounding point (bit 15 of A0 when no-scaling; bit 0 of A1 when scaling down; bit 14 of A0 when scaling up) before the bit truncation during a rounding operation. Figure 3-12 shows the two possible cases.

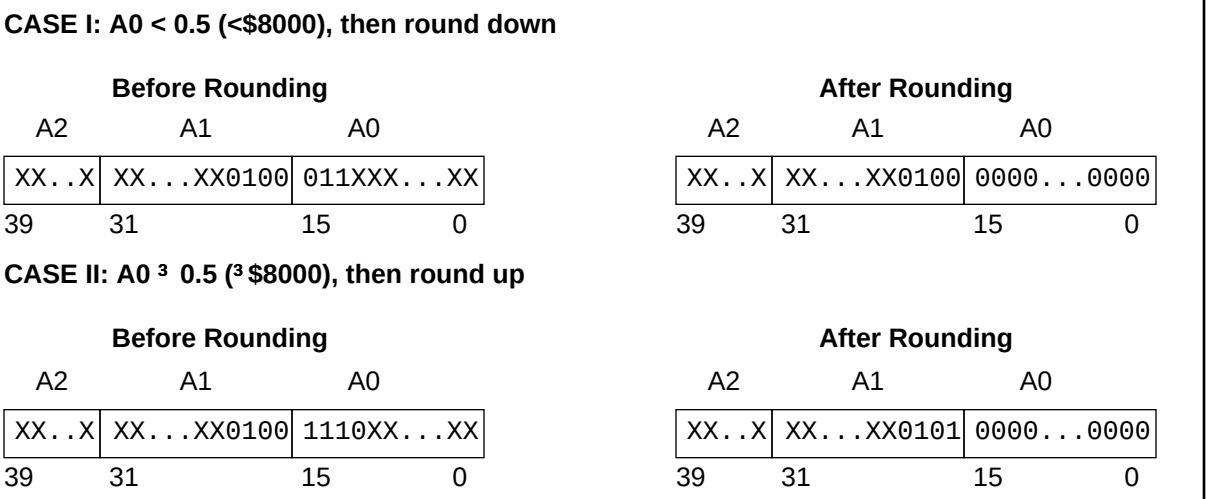


Figure 3-12 Two's Complement Rounding (No-scaling)

Once the rounding bit has been programmed in the OMR, there is a delay of one instruction cycle before the new rounding mode becomes active.