Converting an Unsigned Integer Product to a Signed Integer Product

Consider the problem of finding the product of two signed 2's complement integers. We'll do this using the same approach we used for fixed-point multiplication, i.e., compute the unsigned product and then modify it - except that we need to keep the full double-length result rather than just the middle half. We'll do it here using four-bit operands, but the principle is the same regardless of operand size.

First the math:

$$A_uB_u = (2^3A_3 + A_{2..0})(2^3B_3 + B_{2..0}) = 2^6A_3B_3 + 2^3(A_3B_{2..0} + B_3A_{2..0}) + A_{2..0}B_{2..0}$$

$$A_sB_s = (-2^3A_3 + A_{2..0})(-2^3B_3 + B_{2..0}) = 2^6A_3B_3 - 2^3(A_3B_{2..0} + B_3A_{2..0}) + A_{2..0}B_{2..0}$$

$$= A_uB_u - 2^4A_3B_{2..0} - 2^4 B_3A_{2..0}$$

This equation tells us that for each negative operand, we should subtract the three least-significant bits of the other operand from the most-significant half of the unsigned product. When performing these subtractions, even though the math says to subtract only the three least-significant bits (i.e., $A_{2..0}$ or $B_{2..0}$ or both) from the unsigned product, you can actually use the entire 4-bit operand (A_s or B_s):

$$A_sB_s = A_uB_u - 2^4A_3B_s - 2^4 B_3A_s$$

Case 1 - Both operands are positive: No subtractions are performed.

Case 2 - Operands of different signs: We subtract the positive operand from the most-significant half of the unsigned product. The fourth bit of the positive operand is zero, and thus has no effect on the result.

Case 3- Both operands are negative: The most-significant bits of both 4-bit operands are 1's. The first subtraction thus changes the sign of the result; the second subtraction changes it back – the same final result you would get if the subtraction used only the least-significant three bits (i.e., $A_{2..0}$ and $B_{2..0}$).

Example 1 - Positive × Positive:

		A_s	$+7_{10}$	←		0	1	1	1	→	7_{10}	A_u
		B_s	$+5_{10}$	←	×	0	1	0	1	→	5_{10}	B_u
						0	1	1	1			
					0	0	0	0				
				0	1	1	1					
			0	0	0	0						
		0	**0**	**1**	**0**	**0**	**0**	**1**	**1**	→	35_{10}	A_uB_u
	-2^4A_s	*B≥0, do not subtract A_s*										
		0	0	1	0	0	0	1	1			
	-2^4B_s	*A≥0, do not subtract B_s*										
A_sB_s	$+35_{10}$ ←	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**			

Converting an Unsigned Integer Product to a Signed Integer Product

Example 2a - Positive × Negative:

		A_s	-7_{10}	←		1	0	0	1	→	9	A_u
		B_s	$+5_{10}$	←	×	0	1	0	1	→	5	B_u
						1	0	0	1			
					0	0	0	0				
				1	0	0	1					
			0	0	0	0						
		0	**0**	**1**	**0**	**1**	**1**	**0**	**1**	→	45	A_uB_u
	-2^4A_s	*B≥0, do not subtract A_s*										
		0	0	1	0	1	1	0	1			
	-2^4B_s	0	1	0	1							
A_sB_s	-35_{10}	←	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**			

Example 2b - Negative × Positive:

		A_s	$+7_{10}$	←		0	1	1	1	→	7	A_u
		B_s	-5_{10}	←	×	1	0	1	1	→	11	B_u
						0	1	1	1			
					0	1	1	1				
				0	0	0	0					
			0	1	1	1						
		0	**1**	**0**	**0**	**1**	**1**	**0**	**1**	→	77	A_uB_u
	-2^4A_s	0	1	1	1							
		1	1	0	1	1	1	0	1			
	-2^4B_s	*A≥0, do not subtract B_s*										
A_sB_s	-35_{10}	←	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**		

Example 3 - Negative × Negative:

		A_s	-7_{10}	←		1	0	0	1	→	9_{10}	A_u
		B_s	-5_{10}	←	×	1	0	1	1	→	11_{10}	B_u
						1	0	0	1			
					1	0	0	1				
				0	0	0	0					
			1	0	0	1						
		0	**1**	**1**	**0**	**0**	**0**	**1**	**1**	→	99_{10}	A_uB_u
	-2^4A_s	1	0	0	1							
		1	1	0	1	0	0	1	1			
	-2^4B_s	1	0	1	1							
A_sB_s	$+35_{10}$	←	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**		