Booth Multiplication Algorithm

Abenet Getahun
Fall 2003 CSCI 401

Booth Multiplication Algorithm

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation.

I will illustrate the booth algorithm with the following example:
Example, 2_{ten} x $(-4)_{ten}$
0010_{two} * 1100_{two}

Step 1: Making the Booth table

I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.
i.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change ,so there are two changes on this one
1100 -- From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one.
Therefore, multiplication of 2 x (– 4), where 2_{ten} (0010_{two}) is the multiplicand and $(-4)_{ten}$ (1100_{two}) is the multiplier.

II. Let X = 1100 (multiplier)
Let Y = 0010 (multiplicand)
Take the 2's complement of Y and call it –Y
–Y = 1110

III. Load the X value in the table.

IV. Load 0 for X-1 value it should be the previous first least significant bit of X

V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.

VI. Make four rows for each cycle; this is because we are multiplying four bits numbers.

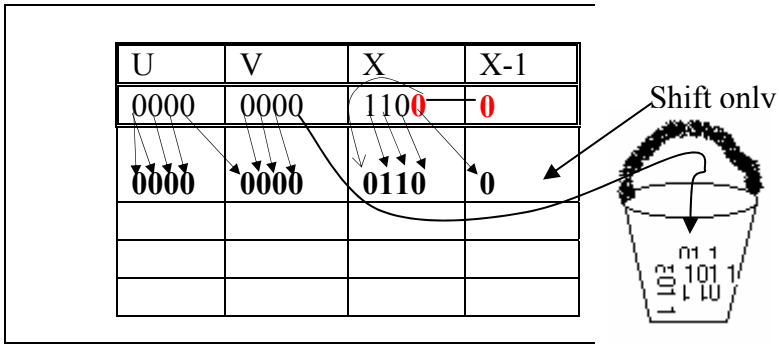
U	V	X	X-1	
0000	0000	1100	0	Load the value
				1st cycle
				2nd cycle
				3rd Cycle
				4th Cycle

Step 2: Booth Algorithm

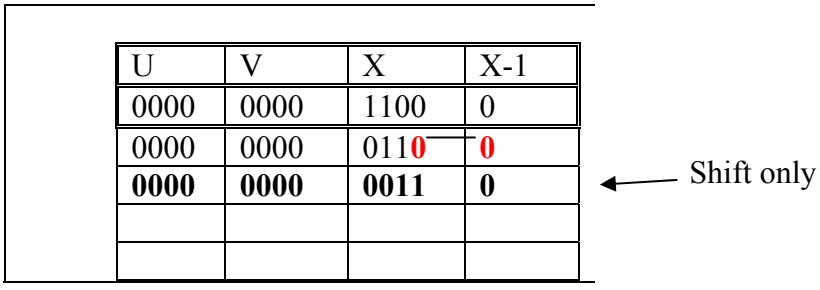
Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtracted from the partial product, or left unchanged according to the following rules:

Look at the first least significant bits of the multiplier "X", and the previous least significant bits of the multiplier "X - 1".

I	0 0	Shift only
	1 1	Shift only.
	0 1	Add Y to U, and shift
	1 0	Subtract Y from U, and shift or add (-Y) to U and shift
II	Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. Thus a positive number remains positive, and a negative number remains negative.	
III	Shift X circular right shift because this will prevent us from using two registers for the X value.	



Repeat the same steps until the four cycles are completed.



U	V	X	X-1	
0000	0000	1100	0	
0000	0000	0110	0	
0000	0000	0011	**0**	
1110	**0000**	**0011**	**0**	← Add –Y (0000 + 1110 = 1110)
1111	**0000**	**1001**	**1**	← Shift

U	V	X	X-1	
0000	0000	*1100*	0	
0000	0000	0110	0	
0000	0000	0011	0	
1110	0000	0011	0	
1111	0000	1001	**1**	
1111	**1000**	***1100***	**1**	← Shift only

We have finished four cycles, so the answer is shown, in the last rows of U and V which is: 11111000_{two}

Note: By the fourth cycle, the two algorithms have the same values in the Product register.