Booth's Algorithms for Multiplication

There is a family of algorithms for speeding multiplication in hardware that are all based on Booth's original work. He observed that a string of 1's in a binary number could be replaced by a carry out with a subtraction at the least significant bit of the string. Put in mathematical terms,

$$\sum_{n=0}^{N-1} 2^n = 2^N - 1$$

As a concrete example, $0111_2 = 1000_2 - 0001_2$. Obviously strings of 1's in the middle of a number can be represented in the same way by left shifting this result appropriately.

This can be exploited in a variety of ways to reduce the number of partial products one has to form and accumulate to calculate a product. In class I talked about a particularly common implementation of this in radix-4 multiplication. The basic idea is to express the multiplier in radix-4, that is, as a set of digits 0 – 3 instead of just 0,1. This cuts the number of partial products in half because you are multiplying by two binary bits at once. However, it requires multiplying by 3 which is difficult. (Multiplication by 0, 1, or 2 is trivial because they only involve simple shifts. Three is the hard one.) To avoid multiplying by 3, we use Booth's observation and recode the digit set to be 2, 1, 0, -1, and -2. The partial products with the positive digits are trivial to form while the negative values can be done by subtracting instead of adding their partial products.

Here is an example: $+6_{10} * +6_{10} = +36$ where the numbers are 4-bit unsigned binary. As needed the negative partial products are extended to a 9-bit 2's complement number. (8-bits is adequate to express any 4 x 4 product unsigned product and the 9th bit allows for worst case sign extension. The 9th bit is lost in the product because the product is never negative. Generally only minor adjustment is needed in this algorithm for 2's complement operands.) Only two partial products are needed and the second is left shifted by two bits because of the radix-4 coding.

```
     0110         - multiplicand = +6
     0110 – multiplier as base 2 = +6
    +2 -2 – multiplier recoded with Booth's  algorithm per table below = +2*4 + (-2) = +6
111110100 – first partial product = -2*(+6) = -12
0001100    - second partial product = +2*(+6) left shift by 2 = +48
 00100100 – final product = +4 + +32 = +36
```

In this table, the bits i and i+1 are the bits being turned into a radix-4 digit and the i-1 th bit contributes to the recoded value for the reasons given in the comment column.

Xi+1	Xi	Xi-1	Zi	Comment
0	0	0	0	0
0	0	1	1	End of string of 1's - carry in
0	1	0	1	Isolated 1
0	1	1	2	End of string of 1's - carry into second bit
1	0	0	-2	Beginning of 1's in 2nd bit
1	0	1	-1	-2 + 1
1	1	0	-1	Beginning of 1's in 1st bit
1	1	1	0	String goes through - all zero