Multiplier Circuits

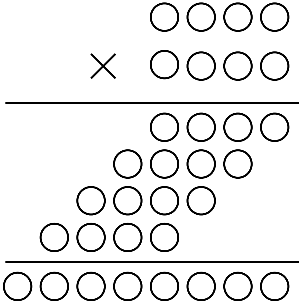
Dinesh Sharma

EE Department
IIT Bombay, Mumbai

October 8, 2018

Shift and Add Multipliers

An obvious way for implementing multipliers is to replicate the paper and pencil procedure in hardware.



- Initialize the product to 0, extend multiplicand to left by n bits filled with 0s.
- If the least significant bit of the multiplier is 1, add the multiplicand to product, else do nothing.
- Shift the multiplier right by one bit.
- Shift the multiplicand left by one bit.
- Repeat for n bits

Shift and Add Multipliers

- Each term being added to form the product is called a partial product.
- The name “partial product” is also used for individual bits of the terms being added - so beware!
- The paper-pencil procedure requires n-1 additions to a 2n bit accumulator.
- This uses a single adder, but takes long to complete the multiplication. A 32 x 32 multiplication will require 31 addition steps to a 64 bit accumulator.
- Multiplication can be made faster by using multiple adders and adding terms in a tree structure.

Array Multiplier

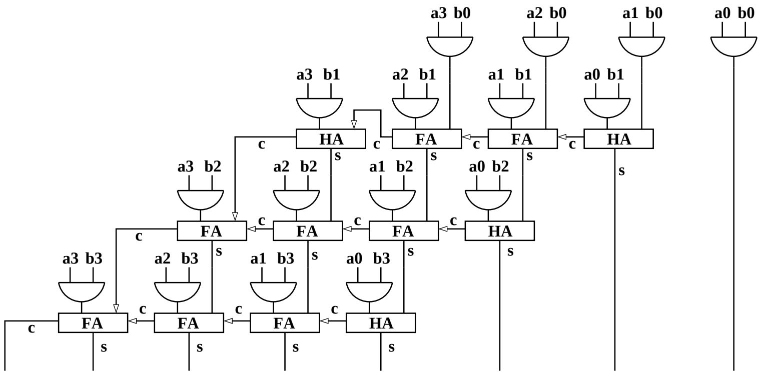
Suppose we want to multiply two n-bit numbers A and B, where

$$A = \sum_{i=0}^{n-1} 2^i a_i \qquad B = \sum_{j=0}^{n-1} 2^j b_j$$

- We can regard all bits of the partial products as an array, whose (i,j)th element is $a_i \cdot b_j$. Notice that each element is just the AND of a_i and b_j.
- **All** elements of the array are available in parallel, within one gate delay of arrival of A and B.
- We can now use an array of full adders to produce the result. One input of each adder is the sum from the previous row, the other is the AND of appropriate a_i and b_j.
- This architecture is called an array multiplier.

Array Multiplier

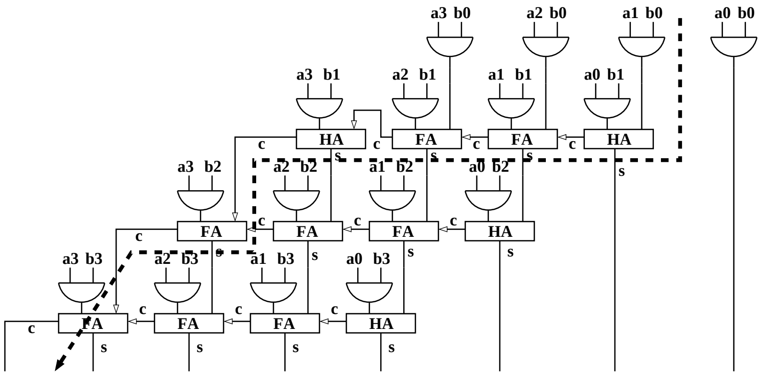
A 4X4 array multiplier is shown below.



Half adders can be used at the right end.

Critical Path through Array Multiplier

The critical path through a 4X4 array multiplier is shown below.



The critical path involves carry as well as sum outputs!

Speeding up Multipliers

The array multiplier has a regular layout with relatively short connections. However, it is still rather slow.

How can we speed up a multiplier?

There are two possibilities:

- Somehow reduce the number of partial products to be added. For example, could we multiply 2 bits at a time rather than 1?
- Since we have to add more than two terms at a time, use an adder architecture which is optimized for this.

Booth Encoding

Booth Encoding reduces the number of partial products by multiplying 2 bits at a time.

- Let the multiplicand be A and the multiplier B.
- Rather than multiplying A with successive bits of B, we can multiply it with two bits of B at a time.
- Depending on the two bits being 00, 01, 10 or 11, the partial product will be 0, A, 2A or 3A.
- 0 and A can be produced trivially.
- 2A can be produced easily by a left shift of A.
- Generating 3A presents a problem!

Booth Encoding

Multiplying the Multiplicand A by 2 bits of the multiplier at a time requires the generation of 0, A, 2A or 3A as partial products. Generating 0, A or 2A is easy.

- 3A cannot be generated directly. However, 3A can be expressed as 4A - A.
- The task of adding 4A is passed on to the next group of 2 bits of the multiplier.
- Since the place value of the next group of 2 bits is 4 times the current one, adding 4A to the product is equivalent to adding 1 to the next group of 2 bits of the multiplier.
- -A can be generated from A, using an adder/subtracter rather than an adder for accumulating the sum of partial products.

Modified Booth Encoding

To simplify the logic for deciding whether an additional 4A should be added on behalf of the less significant 2 bits in the multiplier, we express 2A also as 4A - 2A.

- Since we anyway have an adder-subtracter, this requires no additional resources.
- The modified logic is: for 00, do nothing. For 01, add A.
  for 10, subtract 2A, ask the next group to add 4A.
  for 11, subtract A, ask the next group to add 4A.
- Now the next group can just look at the more significant bit of the previous group and add 1 to the multiplier if it is ‘1’.

Modified Booth Encoding

- The partial product generator looks at the current 2 bits and the MSB of the previous group of 2 bits to decide its action.
- Thus, we scan the multiplier 3 bits at a time, with one bit overlapping.
- For the first group of 2 bits, we assume a 0 to the right of it.
- After handling the previous group, the multiplicand is shifted left by 2 positions. Thus, it has already been multiplied by 4.
- Therefore, adding 4 A on behalf of the previous group is equivalent to adding 1 to the multiplier corresponding to the current group.

Modified Booth Encoding

The following table summarizes the effective multiplier for generating the partial product.

Current 2-bits	Multiplier for these	Previous MSBit	Pending Increment	Total Multiplier
00	0	0	0	0
01	+1	0	0	+1
10	-2	0	0	-2
11	-1	0	0	-1
00	0	1	+1	+1
01	+1	1	+1	+2
10	-2	1	+1	-1
11	-1	1	+1	0

Notice that a 111 in the 3 bit group being scanned requires no work at all.

Modified Booth Encoding

3-bits	Multiplier
000	0
001	+1
010	+1
011	+2
100	-2
101	-1
110	-1
111	0

What happens if there is a string of ‘1’s in the multiplier?

- There will be a -1 in the beginning, because the group begins with 110.
- Similarly, there will be a +1 at the end, because it will end with 011.
- However, for the length of continuous ‘1’s, nothing needs to be done (add zeros).

Thus Booth encoding reduces the number of partial products to half (multiplying 2 bits at a time).

It makes addition in columns of partial products fast because carry propagation during addition will be reduced.